Universidad Carlos III de Madrid

GRADO EN INGENIERÍA DE SISTEMAS AUDIOVISUALES

ESCUELA POLITÉCNICA SUPERIOR

TFE DESARROLLO DE APLICACIÓN DE GESTIÓN PARA PYMES EN TECNOLOGÍA ANDROID

AUTOR: JORGE CABALLERO AMO

TUTOR: MARÍA CELESTE CAMPO VÁZQUEZ

LEGANÉS, 22 DE JUNIO DE 2016

Título: TFE DESARROLLO DE APLICACIÓN DE GESTIÓN PARA PYMES EN TECNOLOGÍA ANDROID

Autor: Jorge Caballero Amo

Director: Carlos Cuadrado Ortega

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 05 de junio de 2016 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Índice

Índice de ilustraciones

Índice de tablas

Capítulo 1. Introduction

This chapter will be a brief English summary of the most important areas of the project, including the reasons why is carried out, the scope and the methodology that has been followed. It also includes the structure that follows the memory at the end of this chapter.

1.1 Project introduction

The main objective of this project is to develop a **mobile application for managing a company's intranet in Android technology**. The target audience is any active company and the final users are its employees. This first prototype is focused in SME (Small Medium Enterprise) but it can be long term scalable.

This project was accomplished in **Armadillo Amarillo S.C.** This company is part of the IT sector with extensive experience in software development and deployment of services.

Armadillo Intranet is an internal product of Armadillo Amarillo for the management of the company's intranet. But it is intended to be a product which can be sold to other companies of the sector.

Armadillo Intranet is the teamwork result of several members of the company. They had a formerly web service composed of various modules and their pertinent backend. This Project grew up of the idea of take this web service to mobile devices, specifically in Android and iOS technology.

My role as an active part of the team consisted in the implementation of the Android mobile application integrated with Android Wear besides the development of several API web services of Armadillo Intranet.

The complete process of the project is divided into the following steps:

- Study and knowledge of the initial system in the company.
- Analysis of the expansion of the given starting system to use it as backend for the mobile applications to develop.
- Design and development of backend services.
- Design and development of the mobile application using Android technology.
- Design and development of the Android wearable application.
- Documentation.

The marked objective of this project is to cover different needs related to the business world. In first place, accelerate the accessibility to the daily company data (such as passwords or documents) and to client information (phone numbers, emails or addresses). On the other hand, makes easier the communication between the employees and the management of the projects through hour assignment.

There is another mobile app for iOS created in parallel to this one and it has also been developed as a TFG in this company.

1.2 Motivation

In Spain, the percentage of adult population that uses a smartphone daily to connect to the Internet is very high, and it is growing more each year. The world's population is quickly adapting to new technologies and the mobile market is very important in this sector.

The smartphone is always with you, which means that the speed you can access the mobile device is much greater than the computer, being the same services which you can find in both.

Today there are many mobile applications to facilitate the daily life to enterprises (communication of employees, the control of working hours, etc.). The aim is to unify all these features in a single application, to waste the minimum time and be more efficient at work.

We should consider the added value given to these services, which are the mobility and the easy access every time. We are trying to achieve an increase of the services usage.

1.3 Project scope

It is proposed to develop a mobile application in Android technology, the integration with the wearable and the implementation of the necessary API services to get the data that give sense to the application, made in PHP technology.

Application score:

- *Principal Panel*: this view will show the last 3 available user resources, the last 3 received messages and the 3 last imputed hours.
- *Resources*: this view will show the different available resources groups for the user.
- *Timesheet*: this view will show a calendar of the current month and a list of the imputed hours in the selected day.
- *Messages*: the view will show the user Inbox and Outbox with a search bar that could filter both messages.
- *Contacts*: this view will show a list of the client's enterprises with the option of get the contact detail.

Wearable score:

- *Notifications*: it will show a message notification in the wearable with the possibility of show the message's detail in the device's screen and reply with a predefined text or voice command.

API score:

- *Services*: it will implement the necessary methods to make possible the communication between the mobile device and the backend.
- *GCM*: it will implement the necessary requests to Google Cloud Messaging to enable the communication and notifications in Android device.

1.4 Methodology and planning

The methodology chosen for the project is **SCRUM [1]**, method of agile software development that divides the stages of development in different Sprints, in our case with a duration of 2 weeks, with a review at the end of each one to see the progress and the status of the project.

The team's work is divided in:

- *Development team*: responsible for delivering the product at the end of each Sprint. People who belong to the team (Android, iOS and backend developers) manage the distribution of the tasks in the most optimal way for the development of the product.
- *Product Owner*: person who represents the customer, in our case, being an inner product of the company, representing the company.
- *Scrum Master*: facilitator of monitoring the methodology and the person who ensure that the product reaches the set objectives. The Scrum Master does not manage the development team, it organizes itself. In this project this role has been carried out by the company's tutor.

The proposed calendar is divided into:

- *Sprint 0*: 1 February - 9 February
- *Sprint 1*: 10 February - 24 February (review 25 February)
- *Sprint 2*: 25 February - 10 March (review 14 March)
- *Sprint 3*: 14 March - 27 March (review March 28)
- *Sprint 4*: 28 March - 11 April (review 12 April)
- *Sprint 5*: 12 April - 26 April (review 27 April)
- *Sprint 6*: 27 April - 11 May (review 12 May)

1.5 Work phases

The project was divided in different phases. At first, a conceptualization phase, design of the application and estimation of the development time which had been done during the Sprint 0. Afterwards, it began the development and testing phase which was the longest phase and had been done during the Sprints 1, 2, 3, 4 and 5. Finally, the writing of the document which had been done in the Sprint 6.

1.6 Hardware and software tools

In this chapter, we are going to make a list of both tools employed to develop this project (hardware and software).

Hardware:

- MacBook Pro (13-inch, early 2011): personal computer development.
  - Processor: 2.3 GHz Intel Core i5
  - Memory: 8 GB 1600 MHz DDR3
  - Graphics: Intel HD 3000 Graphics 512 MB

- Samsung Galaxy Core 2 device: smartphone for development and testing.
  - Processor: 1.2 GHz Quad-Core
  - Screen: 4.5 "(114.2mm)
  - Version Android: 4.4.2 (KitKat)

- LG Watch Urbane: Smart clock for development and testing.
  - Operating system: Android Wear
  - Circular touch screen 1.3 "
  - Compatible with Android 4.3 or higher

Software

- Android Studio 1.5.1: Android application development IDE.
- IntelliJ IDEA 15: IDE development for the backend PHP services.
- Postman: web client to make requests to the API.
- Emulator Android Wear Round 21 API: software to emulate the wearable in the
- computer.
- Genymotion: software to emulate the smartphone in the computer.

1.7 Socio-economic environment

According to the report “Mobile Consumption 2015” elaborated by **Deloitte [2]**, Spain leads penetration of smartphones with 88% of the population (Europe 82%), increasing 3 percentage points in a year and 19 points regarding 2013. This report also reveals that 50% of mobile users consult their devices in the first and last minutes of the day.

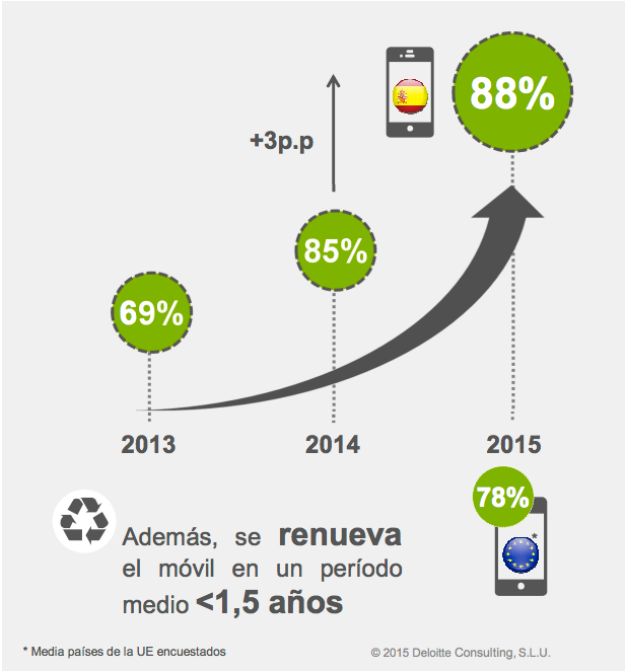

Ilustración 1 – Smartphone penetration in the Spanish market

Mobile technology is changing very quickly the behaviour and habits of the world society, both in stable and advanced markets and in emerging markets in developing countries.

Over the past years, mobile applications have revolutionized the way of living of the global society. Most of the companies are studying making an integration of its business with the mobile phone if they have not developed it already, which generates a strong economic impact in the ICT sector (Information and Communication Technologies).

The success that the mobile applications have in society is due to the facilitator work implanted in users, who can access the information anytime from the menu of their devices.

This trend has generated in the consumer the need to have applications for all the features in their life. If you want to know where to go for dinner you will search for restaurants close to you in a mobile app, if you want to catch a taxi you will ask for it through the app, if you want to check your bank's accounts you will do it through your application, without forgetting the strong impact that generate application focused on communication between people and social networks.

Mobile applications have created the dependence and need in the society of being able to access the information without losing time.

But it is not only focused on facilitating the daily life of society in lighter work, also there is a significant niche oriented mobile technology for health, which can significantly improve the life of the society.

In conclusion, what is intended with this project is to facilitate the life of the employees of any enterprise to make their working time as efficient as possible, increasing the real productivity and generating an economic impact in the company.

1.8 Regulatory Framework: Privacy policy and data processing

In the mobile application explained in this document it is necessary that users enter its personal information like name, surname, ID, email or contact phone. that's the reason why the organic law of data personal data protection is applied.

This law *"[…] tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar.*

Esta ley afecta a todos los datos que hacen referencia a personas físicas registradas sobre cualquier soporte, informático o no. [1] *[…]"*

It also uses the General Telecommunications law. Specifically, we can focus on Article 41. Protection of personal data , which says *"[…] la garantía de que sólo el personal autorizado tenga acceso a los datos personales para fines autorizados por la Ley, la protección de los datos personales almacenados o transmitidos de la destrucción accidental o ilícita, la pérdida o alteración accidentales o el almacenamiento, tratamiento, acceso o revelación no autorizados o ilícitos y la garantía de la aplicación efectiva de una política de seguridad con respecto al tratamiento de datos personales* [2] *[...]"*. It is also used the Article 43. Encryption in networks and electronic communications services, which says *"[...] cualquier tipo de información que se transmita por redes de comunicaciones electrónicas podrá ser protegida mediante procedimientos de cifrado [...]".*

Therefor, the data managed in this application will be collected in a data base in an automatized way under the responsibility of Armadillo Amarillo S.C..

The gathered information can include: general contact data (name, surname, address, contact pone, email…) and particular data (employment information).

Armadillo Amarillo commit to guarantee that the managed information will only be accessible to authorized personal and its use will be limited to the allowed actions allowed from the contracting company.

All the data will be treated as confidential, either the employee's information
as the commercial contacts data from the contracting business. These last will be informed as well of the treatment of its personal data.

In technical terms, the legislation will only affect when it is concerning to the protection of the user's data in security terms. Therefore, there is an encoded communication between the mobile app and the server using Base64. This is a positional numbering system that uses 64 as a base. It is the largest power of two which can be represented using only the ASCII printable characters. The transmission of the information will be on HTTPS (Hypertext Transfer Protocol Secure).

[1] La Ley Orgánica 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal (BOE núm. 298 de 14 de Diciembre de 1999)

[2] Ley 32/2003, de 3 de noviembre, General de Telecomunicaciones. (BOE núm. 264 de 04 de Noviembre de 2003)

1.9 Document structure

Chapter 1, Introduction: there will be a brief summary of the document. It would be analysed the scope of the project, the motivation to carry it out, the methodology used and a breakdown of the project phases.

Chapter 2, State of the art: it will discuss the situation of mobile technologies so far, putting an emphasize on the chosen development environment.

Chapter 3, Requirements specifications and application description: it will carry out an analysis of the functionalities of the application and also there will be detailed the requirements which should be fulfilled. Besides, the work methodology and the estimation of the project will be examined, as well as the context of the former situation (the web application) before beginning developing.

Chapter 4, Application development: it will perform a technical analysis of the development of the application focusing on the diagram and classes structure. Also, there will be an evaluation of the tests that have been conducted to determine the Software quality.

Chapter 5, Conclusions: there will be presented the conclusions drawn after carrying out the project and the possible development future lines.

Capítulo 2. Estado del arte

En este capítulo se expondrán los conceptos necesarios para entender el estado de la tecnología móvil tras una investigación del sector.

Primero, realizaremos un análisis general del mercado móvil para luego centrarnos en el entorno de desarrollo móvil seleccionado. Después, realizaremos un análisis del mercado de desarrollo de aplicaciones móviles relacionadas con nuestro proyecto para ver cuáles son las tendencias actuales.

2.1 Entornos móviles

Dentro del desarrollo móvil podemos diferenciar dos tendencias claras: *desarrollo nativo* y *desarrollo multiplataforma*.

2.1.1 Desarrollo nativo

Las aplicaciones desarrolladas en nativo únicamente sirven para el sistema operativo para el que se implementan. Encontramos varios sistemas operativos dónde podemos destacar **Android**, **iOS** y **Windows Phone**.

Android [3]. Sistema operativo OpenSource desarrollado por Google y basado en Linux. El lenguaje de programación oficial es Java.

iOS [4]. Sistema operativo propiedad de Apple Inc. El lenguaje de programación utilizado es Objetive-C y Swift.

Windows Phone [5]. Sistema operativo desarrollado por Microsoft. El lenguaje de programación utilizado es C# y Visual Basic .NET.

2.1.2 Desarrollo multiplataforma

Las aplicaciones multiplataforma sirven, como indica su nombre, para diversas plataformas. Podemos destacar varios frameworks como Apache Cordova/PhoneGap, AngularJs e Ionic. Todos ellos utilizan HTML5, CSS y Javascript.

Apache Cordova [6]. Framework de desarrollo de aplicaciones móviles creado por Nitobi.

AngularJs [7]. Framework de código abierto desarrollado por Google. Se utiliza principalmente para crear aplicaciones web.

Ionic [8]. Framework de código abierto construido sobre AngularJs y Cordova desarrollado por Drifty.

El desarrollo de aplicaciones móviles en nativo produce mejores resultados cuando se trata de utilizar hardware del propio dispositivo (cámara, GPS, etc.), por lo tanto, como nuestra aplicación móvil va a utilizar estos elementos, desarrollaremos la aplicación en nativo.

Dentro del mercado nativo, existe un duopolio de Android e iOS, pero existen varios estudios de mercado[3] que sitúan Android como el software con mayor instalaciones en todo el mundo, por lo tanto, vemos como mejor opción desarrollar nuestra aplicación en **Android**, para que llegue a la mayor parte del mercado, centrándonos en nuestro caso, los empleados de cualquier empresa nacional e internacional.

Por otro lado, la empresa en la que se realiza este proyecto requería el desarrollo de la aplicación para los dos sistemas operativos mayoritarios del mercado, es por ello que se han realizado dos proyectos en paralelo, centrándose mi papel en el desarrollo Android.

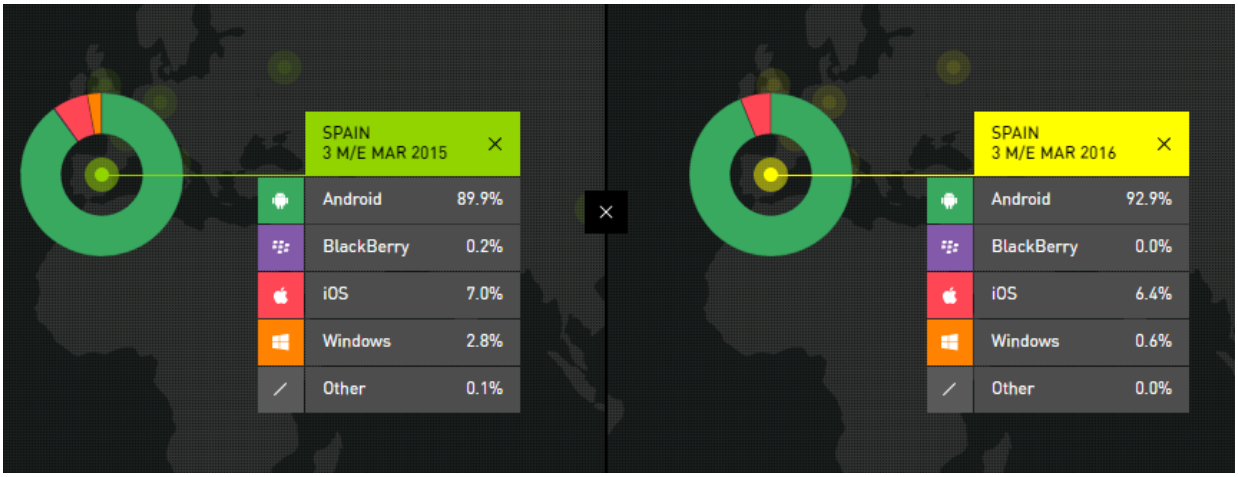

Ilustración 2 - Diferencia del mercado móvil en España entre marzo de 2015 y marzo de 2016.[4]

Como podemos ver en la imagen anterior, Android posee un amplio margen de mercado en el territorio español, aumentando en un año 3 puntos porcentuales a costa de los demás sistemas operativos, principalmente Windows Phone.

[3] Kantar WorldPanel. "Android vuelve a crecer en Europa", *kantarworldpanel.com*, (Diciembre 2015) [en línea]. Disponible en: http://www.kantarworldpanel.com/es/Noticias/Android-vuelve-a-crecer-en-Europa [Consulta: 04 de junio 2016]

[4] Kantar WorldPanel. "Smartphone OS sales market share", *kantarworldpanel.com*, (Diciembre 2015) [en línea]. Disponible en: http://www.kantarworldpanel.com/global/smartphone-os-market-share/ [Consulta: 04 de junio 2016]

2.2 Entorno de desarrollo móvil seleccionado

En este apartado realizaremos una explicación general del sistema operativo Android y la estructura que sigue. Por otro lado, también repasaremos el funcionamiento de Google Cloud Messaging y el entorno de desarrollo del proyecto.

2.2.1 Visión general de Android

En esta sección haremos un repaso histórico del sistema operativo Android hasta llegar al momento actual. Android es un sistema operativo diseñado para dispositivos móviles de pantalla táctil, relojes inteligentes, televisores y automóviles. Las plataformas de Android se identifican mediante su versión, nivel de la API o nombre comercial (código de versión).

El nivel de API se identifica con números enteros y cuanto mayor sea más funcionalidades incluirá. Todas las plataformas lanzadas son retrocompatibles, es decir, solo se añaden funcionalidades con respecto a la API anterior, y si un método ya no se utiliza en la API actual, éste no se elimina si no que se etiqueta como obsoleto (pudiéndose seguir utilizando).

Versión	Código de versión	API	Fecha de lanzamiento
1.0	Base	1	23/07/2008
1.1	Base_1_1	2	09/02/2009
1.5	Cupcake	3	27/04/2009
1.6	Donut	4	15/09/2009
2.0-2.1	Eclair	5-7	26/10/2009
2.2-2.2.3	Froyo	8	20/05/2010
2.3-2.3.7	Gingerbread	9-10	06/12/2010
3.0-3.2.6	Honeycomb	11-13	22/02/2011
4.0.3-4.0.4	Ice Cream Sandwich	14-15	18/10/2011
4.1.x, 4.2.x, 4.3	Jelly Bean	16-18	09/07/2012
4.4	KitKat	19-20	31/10/2013
5.0-5.1	Lollipop	21-22	12/11/2014
6.0	Marshmallow	23	05/10/2015

Tabla 1- Versiones Android[5]

[5] Developer Android. "What is API Level?", *developer.android.com* [en línea]. Disponible en: https://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels [Consulta: 05 de mayo 2016]

Código de versión	Distribución
Froyo	0.1%
Gingerbread	2.2%
Ice Cream Sandwich	2.0%
Jelly Bean	20.1%
KitKat	32.5%
Lollipop	35.6%
Marshmallow	7.5%

Tabla 2 - Versiones de Android que accedieron a Google Play Store[6]

El porcentaje que se muestra en la tabla 2 son los dispositivos que accedieron a Google Play Store recogidos durante un período de 7 días que terminó el 2 de mayo de 2016. Cualquiera de las versiones con una distribución de menos de 0,1% no se muestran.

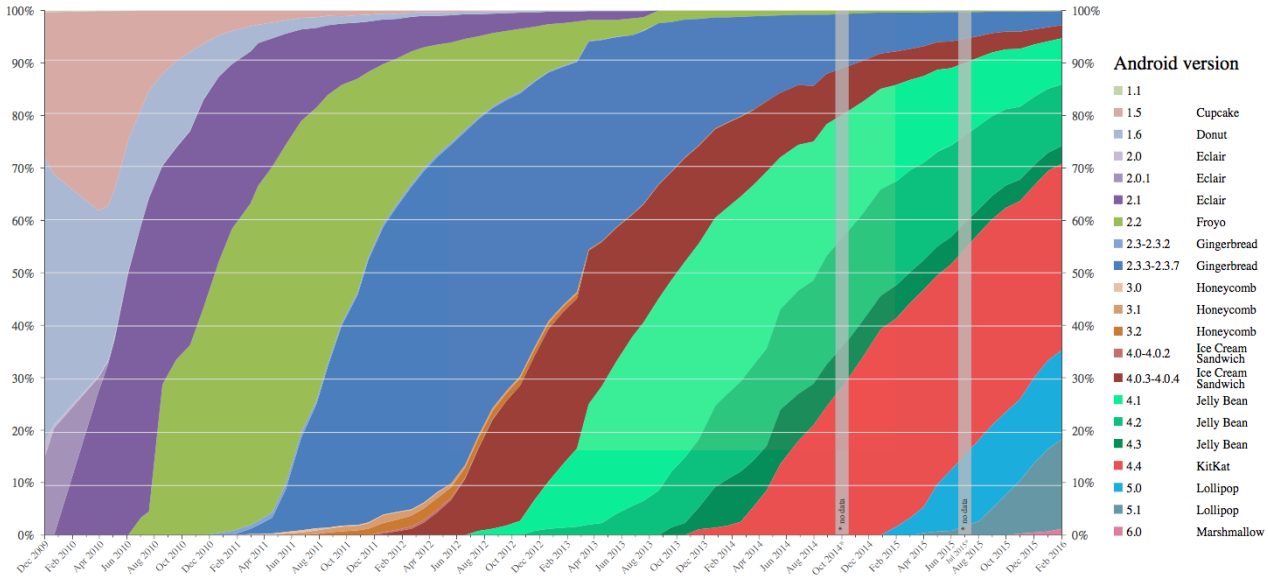

Ilustración 3 - Reparto del mercado dentro del sistema operativo Android[7]

Podemos concluir que la versión más distribuida de Android es **KitKat** con alrededor del 36% del mercado [Ilustración 3], pero la versión que más usuarios activos tiene en el mercado es **Lollipop** con un 35,6% [Tabla 2].

[6] Developer Android. "Platform Versions", *developer.android.com* [en línea]. Disponible en: https://developer.android.com/about/dashboards/index.html?hl=es [Consulta: 05 de mayo 2016]

[7] Wikipedia, *es.wikipedia.org* [en línea]. Disponible en: https://es.wikipedia.org/wiki/Anexo:Historial_de_versiones_de_Android [Consulta: 05 de mayo 2016]

2.2.2 Estructura de Android

Android consta de un núcleo con sistema operativo **Linux** que proporciona acceso al hardware del dispositivo móvil, gestión de la memoria y control de procesos. Se hace uso del **SDK** (Software Development Kit) del cual podemos encontrar una buena documentación en la página oficial de desarrolladores de Android.

El SDK de Android compila el código que se ha generado en lenguaje **Java** en un **APK** (Android package) que es un archivo con extensión *.apk* utilizado para instalar la aplicación en los dispositivos. Otras características del SDK es que permite el acceso al hardware (cámara, GPS, acelerómetro, etc.), tiene soporte nativo de mapas y servicios basados en localización, soporta construir aplicaciones que trabajen en segundo plano, acceso a SQLite (base de datos relacional ligera), implementación de notificaciones y gestión de optimizada de la memoria.

Una aplicación Android se basa en **componentes**, los cuales son **Activities**, **Services**, **Broadcast Receivers** y **Content Providers**.[8] La aplicación se compondrá entonces de uno o más de estos componentes. Todos ellos tendrán que ser declarados en el AndroidManifest.xml.

Activity: controla una única pantalla con una interfaz de usuario. Una aplicación puede tener una o más actividades que relacionadas entre sí construyan una experiencia de usuario satisfactoria y coherente. Las actividades son independientes entre sí, es decir, no es necesario que exista un orden de iniciación. Siempre tiene que existir una activity inicial, que es a la que llama la aplicación al iniciarse.

Services: componente que se ejecuta en segundo plano para realizar operaciones de larga ejecución o para realizar trabajos para procesos remotos. No tiene una interfaz de usuario. Un ejemplo de servicio es la reproducción de música de fondo en una pantalla.

Broadcast Receiver: componente que responde a mensajes de difusión para todo el sistema. Generalmente estos mensajes son enviados por el propio sistema (batería baja, imagen capturada, etc.) pero las aplicaciones también pueden crear sus propios anuncios y avisar a otras aplicaciones. Estos componentes no tienen interfaz de usuario pero normalmente lanzan una notificación para comunicarse con el usuario.

Content Provider: componente que gestiona un conjunto de datos compartido por otras aplicaciones. A través de este componente y con los permisos necesarios, una aplicación puede acceder a la información almacenada en otra e incluso modificar los datos.

Los componentes Activities, Services y Broadcast Receivers se activan mediante **Intents**. Un Intent es una descripción abstracta de una operación que se va a realizar. Se puede lanzar como *startActivity* para Activities, *broadcastIntent* para enviar datos a los Broadcast Receivers y como

[8] Developer Android. "App Components", *developer.android.com* [en línea]. Disponible en: https://developer.android.com/guide/components/index.html?hl=es [Consulta: 05 de mayo 2016]

startService o *bindService* para los Services. El componente Content Provider se lanza a través de *contentResolver*.

Dentro de los Actvities encontramos otro componente de la aplicación denominado **Fragment**, que representa un comportamiento o una porción de la interfaz de la Activity. Se utilizan comúnmente combinando varios Fragments dentro de una Activity para generar una interfaz de usuario con varias vistas o para reutilizar la vista del Fragment en varias Activities.

También se utiliza usualmente los **Adapters**, que son objetos que hacen de enlace entre los datos que proveen a la aplicación y los elementos gráficos de la misma, proporcionando acceso a los elementos de datos.

Otra clase que se merece hacer mención es la clase **AsyncTask**. Esta clase permite realizar operaciones en segundo plano para no afectar al hilo principal de la actividad y evitar una terminación abrupta de la aplicación y no perjudicar la experiencia de usuario. Tiene varios métodos que utilizamos para dar feedback al usuario de que se está realizando la petición:

onPreExecute(): se ejecuta antes de realizar la petición. En nuestro proyecto hemos incluido un diálogo de progreso para indicar al usuario que se ha comenzado a realizar la petición.

doInBackground: construye el JSON para realizar la petición y realiza la misma. En nuestro proyecto es aquí dónde diferenciamos la petición para iniciar sesión, cerrar sesión u otro servicio (que incluirá todos los servicios como recuperar mensajes, información de usuario, recursos, actualizar recurso, enviar mensaje nuevo, etc.).

onPostExecute(): se ejecuta al finalizar la petición, devolviendo a la clase desde dónde ha sido llamado la información necesaria para pintar las vistas.

Todos los componentes principales de una aplicación deben estar inicializados explícitamente en el manifiesto de la aplicación, **AndroidManifest.xml**. A parte, en el manifiesto también se incluye:

- Los permisos que necesita la aplicación para poder ejecutarse (permiso para acceder a Internet o a la memoria del dispositivo).
- El mínimo nivel de la API para la ejecución de la aplicación (no se podrá ejecutar en dispositivos con una API menor).
- Características de hardware y software como el acceso a la cámara del dispositivo, pantalla multitáctil o a servicios bluetooth.
- Las librerías externas que hace uso la aplicación que no sean propias de Android.

2.2.3 Google Cloud Messaging

Google Cloud Messaging (GCM) es un servicio móvil desarrollado por Google que permite a los desarrolladores de aplicaciones enviar notificaciones push con información de sus servidores. Está disponible para los desarrolladores de forma gratuita.

Ilustración 4 – Conexión GCM cliente [9]

En este apartado explicaremos como se establece conexión con **Google Cloud Messaging** de manera general y el flujo de datos que conlleva.

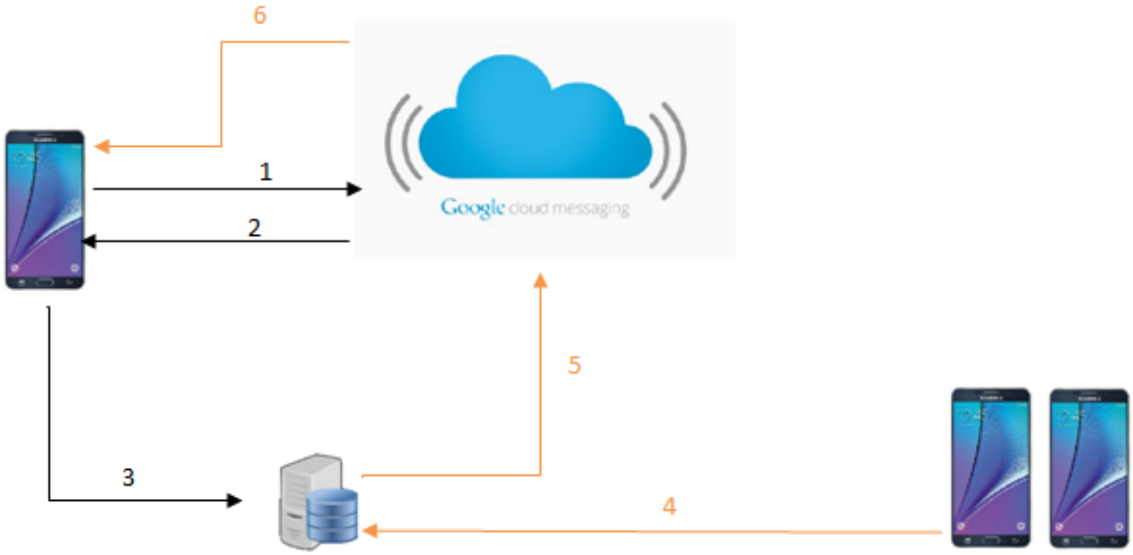

Ilustración 5 – Funcionamiento general Google Cloud Messaging

1. Activación notificaciones push mediante el envío del *sender_ID* del proyecto al que nos queremos dar de alta.

2. GCM registra el dispositivo en su servidor y le asigna un *Registration_token.* Devuelve el *registration_token* al dispositivo.

[9] Console Developer. "Google Cloud Messaging", *console.developers.google.com* [en línea]. Disponible en: https://console.developers.google.com [Consulta: 05 de mayo 2016]

3. El dispositivo se comunica con el su servidor para indicarle cual es el *registration_token* que le ha asignado GCM. Ahora ya tenemos un token relacionado con un dispositivo en concreto.

4. Otro dispositivo envía un mensaje a nuestro dispositivo.

5. Nuestro servidor comprueba si el dispositivo destinatario tiene un *registration_token* que lo identifique como cliente GCM. Si es así, el servidor hace una llamada http (método POST) indicándole en la cabecera la API key para poder comunicarse con GCM, el *registration_token* al que se quiere enviar la notificación y el mensaje.

6. GCM connection server almacena en la cola de notificaciones todos los mensajes para el dispositivo con ese *registration_token*. Comprueba si ese usuario está activo, si es así le hace llegar los mensajes, si no es así los almacena y se los hará llegara cuando el dispositivo se encuentre activo.

Por este motivo de gestión de colas de mensajes de GCM nunca se debe hacer "*unregister*" de GCM, si el usuario quiere dejar de recibir notificaciones por un tiempo lo que se tiene que hacer es borrar de la base de datos de nuestro servidor la relación del usuario con el *registration_token*. De esta manera cuando se vuelva a dar de alta en GCM no se perderá ningún mensaje.

2.2.4 Entorno de desarrollo móvil integrado

En el desarrollo de aplicaciones móviles para Android, el IDE (*Integrated Development Environment)* que se usaba era **Eclipse**, ya que no existía ningún entorno oficial facilitado por Google. En diciembre de 2014 Google anuncia el lanzamiento de **Android Studio 1.0** basada en IntelliJ (IDEA Java IDE) como entorno de desarrollo oficial para Android.

Android Studio es un software libre y está disponible para Windows, MAC OS X y Linux. Google ha ido liberando versiones desde que se lanzó en 2014. Actualmente la versión estable de **Android Studio es la 2.1.1**[10] (liberada el 11 de mayo de 2016), anterior a esta estaba la versión 1.5 (liberada el 17 de noviembre de 2015).

Las diferencias más importantes entre estas dos versiones son:

1. **Instant Run**: Faster Build & Deploy.
2. **Emulador mejorado** con la posibilidad de imitar la configuración de tu móvil de manera muy real y la posibilidad de instalar un .apk arrastrando encima del emulador.
3. **GPU** (Unidad de procesamiento gráfico) novedoso.
4. Está basado en **IntelliJ 15**.

Para la realización de este proyecto se ha utilizado **Android Studio 1.5.1**, ya que cuando se comenzó el desarrollo era la versión estable en el mercado.

[10] Developer Android. "Android Studio", *developer.android.com* [en línea]. Disponible en: https://developer.android.com/studio/index.html [Consulta: 16 de mayo 2016]

2.3 Análisis de aplicaciones de gestión de empresas

Realizamos un estudio de mercado contemplando las diferentes aplicaciones existentes parecidas o que abarcan las funcionalidades que en nuestra aplicación se pretenden desarrollar. Podemos hacer la búsqueda específica de aplicaciones separándolas por módulos, en nuestro caso: *control de horas*, *gestión de clientes*, *comunicación* y *gestión de recursos*.

2.3.1 Control de horas

Existen varias aplicaciones móviles en el mercado para gestionar las horas que se imputan a cada proyecto dentro de las empresas que se dedican al desarrollo software. Dentro del amplio mercado, entramos a analizar más a fondo las siguientes:

Sesame [9]. Herramienta que permite la gestión de los horarios de la empresa permitiendo calcular la rentabilidad de los proyectos conforme a las horas repercutidas a cada uno.

Funcionalidades principales:

- Registrar entradas y salidas de empleados.
- Registrar tiempos dedicados a la ejecución de tareas.
- Solicitar y aceptar peticiones de vacaciones y días libres.
- Ofrecer información y estadísticas a empleadores y empleados de forma sintética.
- Permitir la justificación de horas de dedicadas al trabajo desde fuera de la oficina a través de la aplicación móvil.

Instalaciones: 1.000 - 5.000

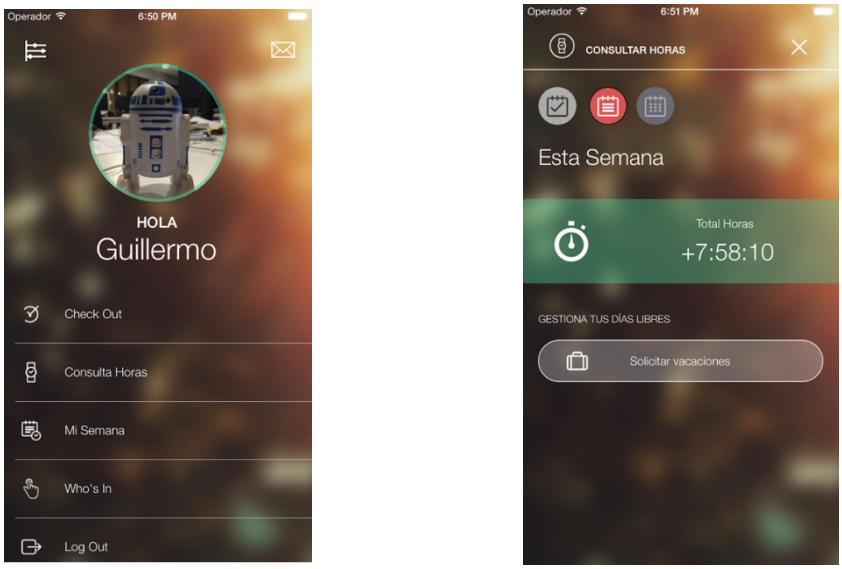

Ilustración 6 – Captura de pantalla Sesame

Intratime [10]**.** Herramienta que permite controlar las entradas y salidas, las horas que trabajas al día y hace uso del servicio de geolocalización para que el usuario pueda saber dónde estuvo en un cierto día a una cierta hora.

Instalaciones: 5.000 - 10.000

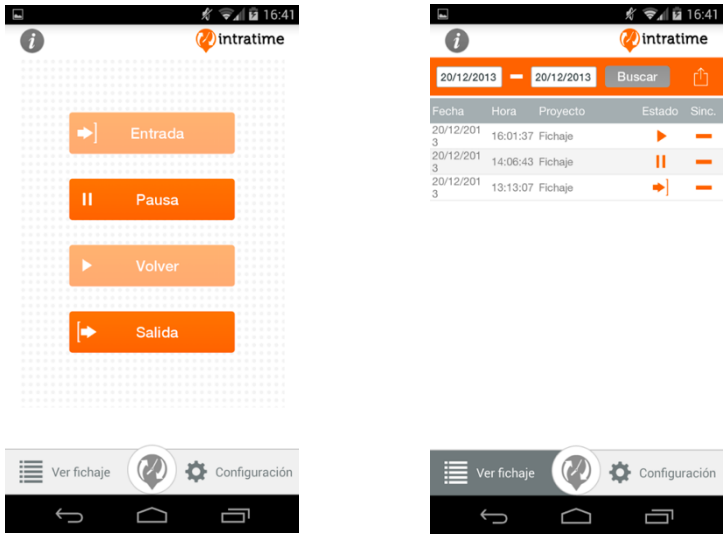

Ilustración 7 – Captura de pantalla Intratime

ClearFocus [11]. Herramienta más orientada a la productividad individual del empleado basada en la técnica Pomodoro, que divide el tiempo de trabajo en intervalos de 25 minutos separados por pausas.

Instalaciones: 100.000 - 500.000

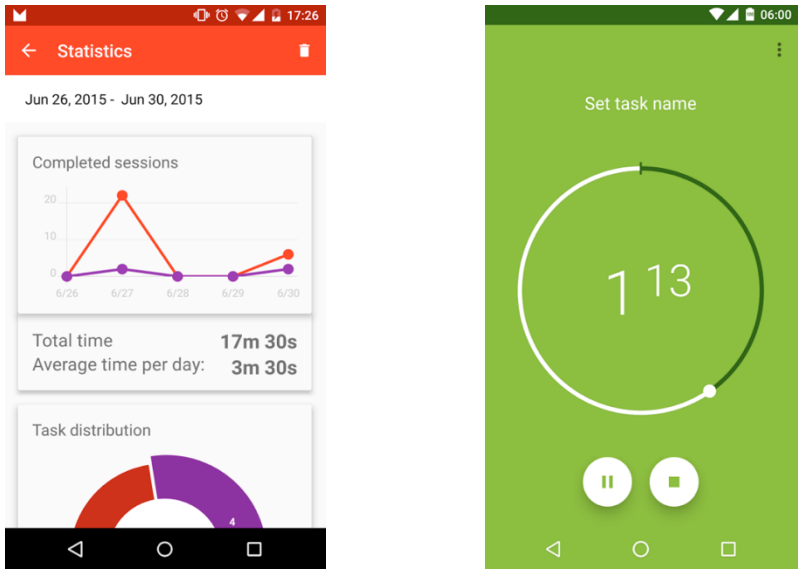

Ilustración 8 – Captura de pantalla ClearFocus

2.3.2 Gestión de clientes

Existen en el mercado varias aplicaciones para gestionar la información y las relaciones con los clientes. Podemos destacar:

Insightly [12]**.** Herramienta muy completa para la gestión de contactos en todas las etapas del proceso de ventas, desde clientes potenciales, eventos, correos u oportunidades de venta.

Funcionalidades principales:

- Crear y actualizar perspectiva e información al cliente.
- Definir fácilmente las relaciones entre los contactos a través de la vinculación.
- Encontrar rápidamente la información que necesita a través de la búsqueda y filtrado.
- Añadir notas de voz.

Instalaciones: 50.000 - 100.000

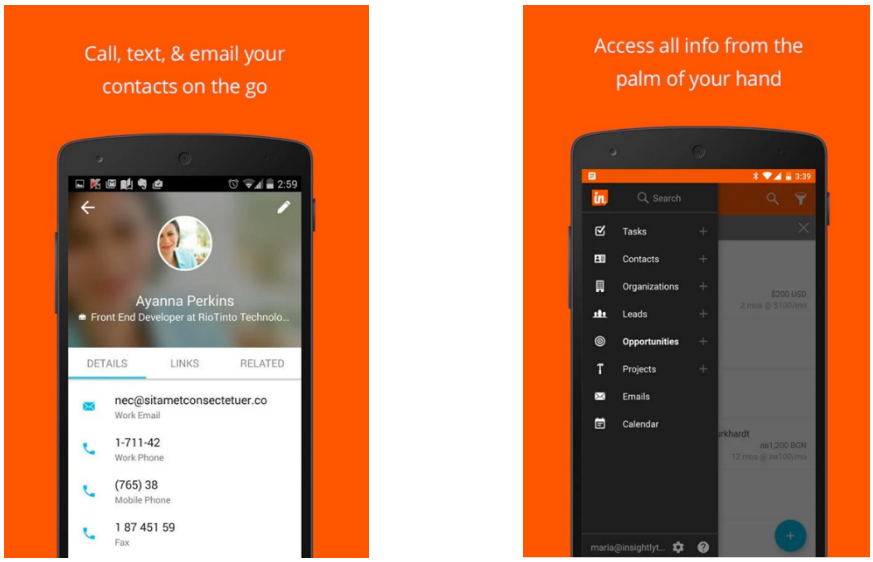

Ilustración 9 – Captura de pantalla Insightly

2.3.3 Comunicación

Existen aplicaciones móviles para facilitar la comunicación entre los trabajadores de una misma empresa con el objetivo de agilizar cualquier trámite y aumentar la productividad. Podemos destacar:

Hipchat [13]**.** Herramienta dedicada a la comunicación entre trabajadores desarrollada por Atlassian, con lo que ello significa: integración con los productos Atlassian como Jira, Jira ServiceDesk, Confluencia y Bitbucket.

Funcionalidades principales:

- Colaborar con los compañeros de trabajo, compañeros de equipo y amigos en torno a los nuevos proyectos, campañas e ideas.
- Mensajería en tiempo real y el uso compartido de archivos hace que sea fácil para su equipo para estar conectado en cualquier lugar.
- Crear un número ilimitado de grupos y conversaciones individuales.
- Compartir y ver archivos e imágenes.
- Integración con Twitter, MailChimp, Wunderlist, UserVoice, Drive, Hangouts

Instalaciones: 100.000 - 500.000

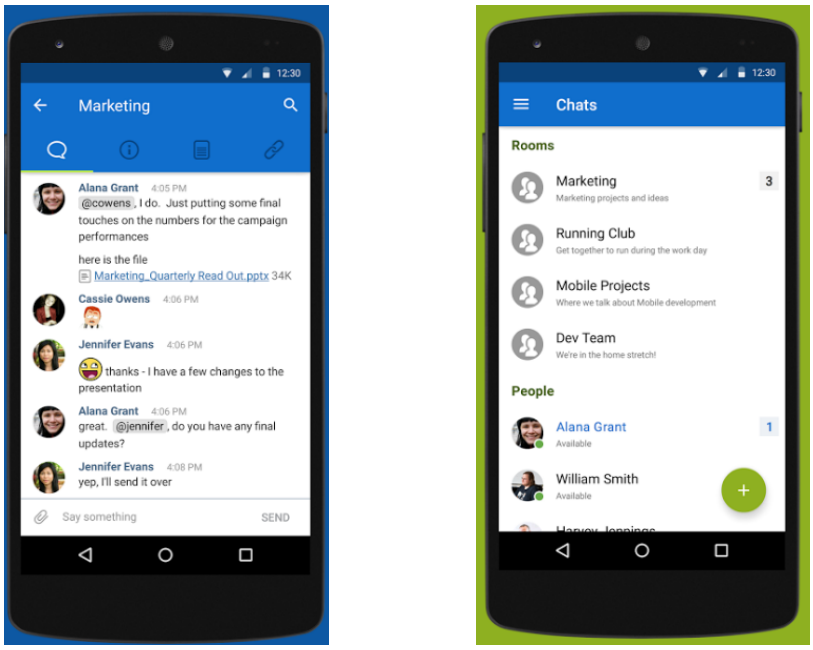

Ilustración 10 – Captura de pantalla Hipchat

Yammer [14]**.** Red social empresarial para colaborar con tu equipo de trabajo y mantenerte conectado.

Funcionalidades principales:

- Colaborar con tu equipo en tiempo real.
- Compartir fotos y vídeos.
- Notificaciones.
- Traducción integrada en 25 idiomas.

Instalaciones: 1.000.000 - 5.000.000

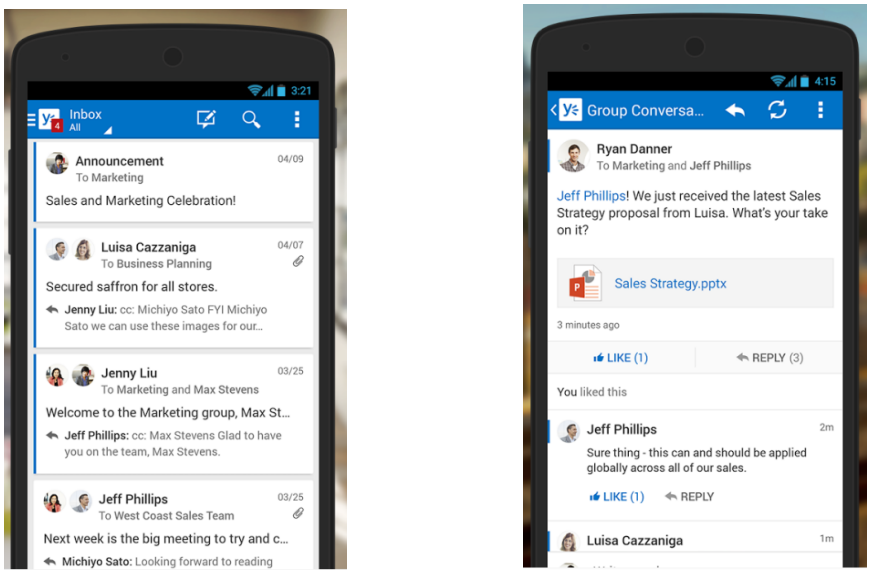

Ilustración 11 – Captura de pantalla Yammer

2.3.4 Gestión de recursos

Lo más parecido que encontramos en el mercado de aplicaciones móviles son para la gestión de contraseñas. Destacamos:

LastPass [15]**.** Herramienta dedicada a la gestión de contraseñas como un llavero virtual en tu dispostivo móvil.

Funcionalidades principales:

- Gestor de contraseñas.
- Autocompletar contraseñas.
- Finger Scan.
- Compartir contraseñas.

Instalaciones: 1.000.000 – 5.000.000

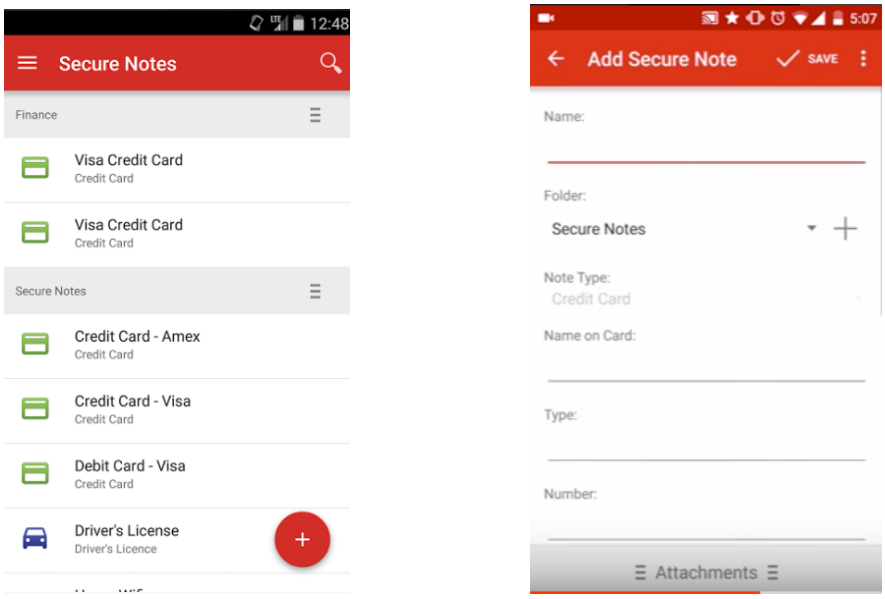

Ilustración 12 – Captura de pantalla LastPass

2.4 Conclusiones

Vemos que la gestión de la información es un campo importante y que se le da mucha importancia dentro del sector TIC. Es por ello que vemos que muchas empresas del sector apuestan por el desarrollo de este tipo de aplicaciones para sus propias empresas y para poder ofrecer un producto a sus posibles clientes.

Analizando las aplicaciones móviles del mercado relacionadas con el sector de desarrollo de nuestro proyecto, podemos observar que existen varias que atacan necesidades específicas, ya sean la gestión de horas y recursos, la comunicación de un equipo y la organización de la información de los clientes. Pero no existe ninguna que abarque todos estos campos al mismo tiempo.

Es en este punto dónde nos vamos a enfocar. Lo que se pretende con este proyecto es la **centralización**, diferenciándose del resto de aplicaciones del mercado brindando la posibilidad de tener todos los datos de la empresa en una misma aplicación, con el objetivo de facilitar a los empleados y a la propia empresa el acceso a la información.

Capítulo 3. Especificación de requisitos y descripción de la aplicación

Antes de comenzar con el desarrollo de la aplicación Android se ha procedido al análisis de la funcionalidad del servidor web de la compañía.

3.1 Contexto

A continuación se exponen os resultados del análisis y un breve resumen de la funcionalidad de las diferentes partes de las que se partía en el proyecto.

3.1.1 Servidor web

Armadillo Amarillo disponía de un servidor web que abarca todos los módulos de la Intranet para los empleados y para el administrador de la empresa desarrollado en PHP. La Intranet orientada al empleado se compone por:

- *Registro Horas.*
- *Mensajes.*
- *Recursos.*
- *Encuestas.*
- *Documentos.*

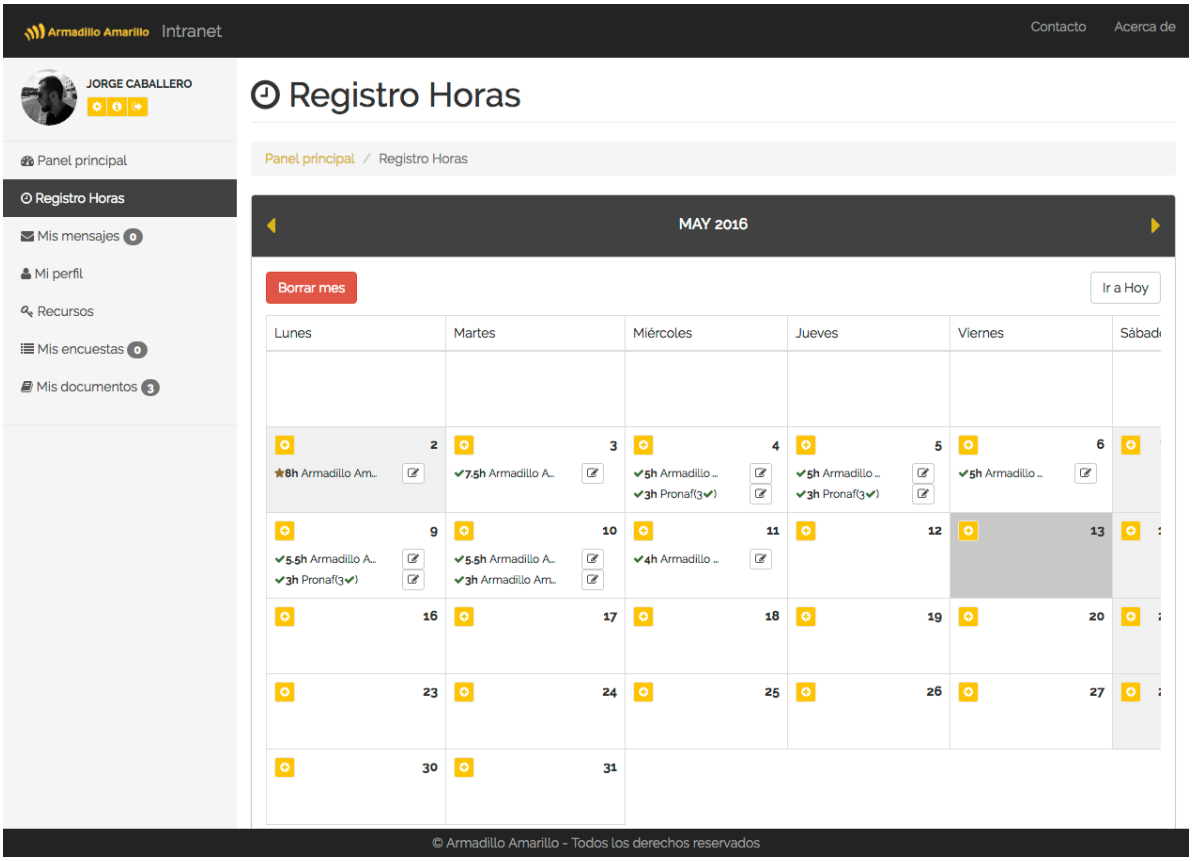

Ilustración 13 – Intranet Armadillo Amarillo – Versión del empleado

La Intranet para el administrador de la empresa añade un panel de administración con módulos adicionales:

- *Comercial*: módulo que gestiona toda la información relacionada con las empresas con las que se ha establecido alguna relación empresarial, los contactos pertenecientes a dichas empresas y la información y el estado de los *leads* de la empresa.

- *Usuarios*: módulo que administra los usuarios de la empresa. Se podrá dar de alta y baja empleados, editar los roles de los usuarios, gestionar el horario, horas y vacaciones, y dar diferentes permisos de administración a los mismos.

- *Recursos*: módulo para gestionar los recursos y grupos de recursos, dar permisos y acceso a los empleados a diferentes tipos de recurso.

- *Selección y RRHH*: módulo que permite ver los procesos de selección que se están llevando a cabo en la empresa y generar nuevas ofertas de empleo.

Ilustración 14 – Intranet Armadillo Amarillo – Versión del administrador

3.1.2 Backend

Armadillo Amarillo disponía de una base de datos en MySQL dónde se almacena toda la información referente a la Intranet. Dicha base de datos se compone de 71 tablas relacionadas entre sí como podemos ver en el Anexo II.

3.1.3 Aplicación móvil previa

Anteriormente, se había inicado el desarrollo un cliente móvil en tecnología Cross, concretamente con Ionic Framework. La funcionalidad era muy básica, se había implementado el login y una opción para aumentar la producción mediante el método Pomodoro de gestión de tiempo. Es por ello que existían varios servicios ya implementados en la API.

3.1.4 API

Exisitían los servicios de login (`initSession`) y logout (`closeSession`).

initSession: petición por POST. Servicio utilizado para iniciar sesión en la aplicación mediante los parámetros de entrada *email* y *password*. El servicio devuelve los datos del usuario si se ha realizado correctamente o algún tipo de error detallado si éste ocurriese.

closeSession: petición por POST. Servicio utilizado para cerrar sesión en la aplicación mediante el parámetro de entrada *auth*, que se trata de un código de seguridad necesario para realizar cualquier petición al servidor.

3.2 Requisitos de la aplicación

A continuación se muestra el modelo de la tabla de requisitos por el que especificarán los requisitos del proyecto.

Título	<Título>	Tipo	<Tipo>
Id ref.	<Identificador>	Épica	<Épica a la que pertenece>
Descripción	<Breve texto explicativo>		
Bloqueada por	<Tareas que bloquean a la incidencia actual>	Bloqueante de	<Tareas que son bloqueadas por la incidencia actual>
Fecha creación	dd/mm/aaaa	Fecha realización	dd/mm/aaaa

Tabla 3 - Plantilla de tabla de requisitos

Título: nombre descriptivo de la tarea.

Tipo: historia de usuario (HU) o Épica.

HU – tarea breve que pertenece a una épica.

Épica – tarea que engloba varias historias de usuario.

Ejemplo: la épica Login/Registro engloba las historias de usuario: diseño de Login, funcionalidad Login y gestión de errores del Login.

Id referencia: identificador único que identifica una historia de usuario.

Épica: nombre de la épica a la que pertenece si es que pertenece a alguna.

Descripción: texto explicativo que define los puntos que se tienen que realizar en una tarea.

Bloqueada por: tareas que bloquean la historia de usuario en la que nos encontramos.

Bloqueante de: tareas que no se pueden empezar hasta que no se termine la historia de usuario en la que nos encontramos.

Fecha de creación: fecha en la que se crea la incidencia en la planificación del proyecto.
Fecha de realización: fecha en la que se termina la historia de usuario.

Los requisitos de la aplicación se dividen en Sprints como se detalla e las secciones siguientes:

3.2.1 Sprint 1

Título	Login/Registro	Tipo	Épica
Id ref.	TFG-1	Épica	-
Descripción	-		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	-

Tabla 4 - Requisito TFG-1 – Login/Registro

Título	Diseño Login	Tipo	HU
Id ref.	TFG-4	Épica	Login/Registro
Descripción	* Logotipo de la empresa. * *Label* "*User*" con *TextEdit*. * *Label* "*Password*" con *TextEdit* con formato *password*. * Botón "*Login*".		
Bloqueada por	-	Bloqueante de	TFG-5
Fecha creación	05/02/2016	Fecha realización	26/02/2016

Tabla 5 - Requisito TFG-4 – Diseño Login

Título	Hacer login	Tipo	HU
Id ref.	TFG-5	Épica	Login/Registro
Descripción	* Conexión introduciendo usuario y *password*.		
Bloqueada por	TFG-4	Bloqueante de	TFG-6
Fecha creación	05/02/2016	Fecha realización	26/02/2016

Tabla 6 - Requisito TFG-5 – Hacer Login

Título	Gestión de errores Login	Tipo	HU
Id ref.	TFG-6	Épica	Login/Registro
Descripción	* Probar qué ocurre si dejamos algún campo vacío, los dos vacíos, contraseña incorrecta o usuario inexistente. * Fallo de conexión para toda la aplicación.		
Bloqueada por	TFG-5	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	26/02/2016

Tabla 7 - Requisito TFG-6 – Gestión de errores Login

Título	Menú Lateral	Tipo	Épica
Id ref.	TFG-10	Épica	-
Descripción	-		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	-

Tabla 8 - Requisito TFG-10 – Menú Lateral

Título	Diseño Menú Lateral	Tipo	HU
Id ref.	TFG-11	Épica	Menú Lateral
Descripción	* Generar vista de diseño de Menú Lateral.		
Bloqueada por	-	Bloqueante de	TFG-12
Fecha creación	05/02/2016	Fecha realización	26/02/2016

Tabla 9 - Requisito TFG-11 - Diseño Menú Lateral

Título	Funcionalidad Menú Lateral	Tipo	HU
Id ref.	TFG-12	Épica	Menú Lateral
Descripción	* Cambiar de vista según ítem seleccionado.		
Bloqueada por	TFG-11	Bloqueante de	TFG-15, TFG-20, TFG-29, TFG-42, TFG-51, TFG-56, TFG-114
Fecha creación	05/02/2016	Fecha realización	26/02/2016

Tabla 10 - Requisito TFG-12 – Funcionalidad Menú Lateral

Título	Panel Principal	Tipo	Épica
Id ref.	TFG-15	Épica	-
Descripción	-		
Bloqueada por	TFG-12	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	-

Tabla 11 - Requisito TFG-15 – Panel principal

Título	Diseño Panel Principal	Tipo	HU
Id ref.	TFG-16	Épica	Panel Principal
Descripción	* Generar vista de diseño de Panel Principal, que incluye vista de los últimos tres registros de horas, mensajes y recursos. * Botón añadir al final de cada título.		
Bloqueada por	-	Bloqueante de	TFG-17
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 12 - Requisito TFG-16 – Diseño Panel Principal

Título	Funcionalidad Panel Principal	Tipo	HU
Id ref.	TFG-17	Épica	Panel Principal
Descripción	* Muestra los últimos tres registros de horas, mensajes y recursos en una tabla. Si no existe ningún registro, mostrar "No existe ningún registro". * Cuando pulsamos encima de cada ítem que te lleve a la vista seleccionada.		
Bloqueada por	TFG-16, TFG-88	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	04/04/2016

Tabla 13 - Requisito TFG-17 – Funcionalidad Panel Principal

Título	Registro de Horas	Tipo	Épica
Id ref.	TFG-20	Épica	-
Descripción	-		
Bloqueada por	TFG-12	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	-

Tabla 14 - Requisito TFG-20 – Registro de Horas

Título	Diseño Registro de Horas	Tipo	HU
Id ref.	TFG-21	Épica	Registro de Horas
Descripción	* Calendario de horas imputadas * Un botón "Ir a hoy", que te lleva al día actual.		
Bloqueada por	-	Bloqueante de	TFG-24
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 15 - Requisito TFG-21 – Diseño de Registro de Horas

Título	Diseño Editar/Añadir Imputación	Tipo	HU
Id ref.	TFG-22	Épica	Registro de Horas
Descripción	* *Label* "Proyecto: " y un *Select*. * *Label* "Horas: " y un *TextEdit*. * *Label* "Razón: " y un *Select*. * *Label* "Descripción: " y un *TextEdit*. * Botón "Cancelar". * Botón "Guardar ".		
Bloqueada por	-	Bloqueante de	TFG-24
Fecha creación	05/02/2016	Fecha realización	26/02/2016

Tabla 16 - Requisito TFG-22 – Diseño Editar/Añadir Imputación

Título	Funcionalidad Registro de Horas	Tipo	HU
Id ref.	TFG-24	Épica	Registro de Horas
Descripción	* Mostrar un listado con las horas imputadas dependiendo del día seleccionado. * Mostrar la vista "Añadir/Editar Imputación" cuando pulsas el botón "Editar". * Mostrar pop-up de confirmación si pulsas botón "Eliminar". * Mostrar la vista "Añadir/Editar Imputación" cuando pulsas el botón "Añadir".		
Bloqueada por	TFG-21, TFG-22	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 17 - Requisito TFG-24 – Funcionalidad Registro de Horas

Título	Modificación peticiones backend	Tipo	HU
Id ref.	TFG-88	Épica	Panel Principal
Descripción	* Modificar API backend para devolver los últimos tres registros y mensajes del usuario.		
Bloqueada por	-	Bloqueante de	TFG-17
Fecha creación	05/02/2016	Fecha realización	04/04/2016

Tabla 18 - Requisito TFG-88 – Modificación peticiones backend

Título	Funcionalidad MVC	Tipo	Épica
Id ref.	TFG-92	Épica	-
Descripción	* Desarrollar estructura Modelo Vista Controlador de la aplicación.		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	04/04/2016

Tabla 19 - Requisito TFG-92 – Funcionalidad MVC

3.2.2 Sprint 2

Título	Recursos	Tipo	Épica
Id ref.	TFG-42	Épica	-
Descripción	-		
Bloqueada por	TFG-12	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	-

Tabla 20 - Requisito TFG-42 – Recursos

Título	Diseño Recursos	Tipo	HU
Id ref.	TFG-43	Épica	Recursos
Descripción	* Muestra el listado de recursos del grupo. * Muestra listado de los subrecursos del grupo específico con un botón de "Añadir".		
Bloqueada por	-	Bloqueante de	TFG-44
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 21 - Requisito TFG-43 – Diseño recursos

Título	Funcionalidad Recursos	Tipo	HU
Id ref.	TFG-44	Épica	Recursos
Descripción	* Pulsando encima del recurso de grupo muestra listado de recursos. * Pulsando encima del recurso específico muestra los datos del recurso para poder editarlo o verlo. * Pulsando encima de añadir recurso muestra un formulario en blanco donde poder insertar los datos del nuevo recurso.		
Bloqueada por	TFG-43	Bloqueante de	TFG-45
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 22 - Requisito TFG-44 – Funcionalidad Recursos

Título	Diseño Crear/Ver/Editar Recurso	Tipo	HU
Id ref.	TFG-45	Épica	Recursos
Descripción	* Muestra formulario con el título, Url, username, password, comentario, n_resource_type (select), n_resource_group (select). * Muestra botones: # Vista editar: Eliminar, Aceptar y Cancelar. # Vista crear: Aceptar y Cancelar.		
Bloqueada por	TFG-44	Bloqueante de	TFG-46
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 23 - Requisito TFG-45 Diseño Crear/Ver/Editar Recurso

Título	Funcionalidad Crear/Ver/Editar Recurso	Tipo	HU
Id ref.	TFG-46	Épica	Recursos
Descripción	* Pulsando Cancelar vuelve atrás sin guardar los cambios. * Pulsando Eliminar elimina el recurso seleccionado. * Pulsando Aceptar crea/modifica el recurso seleccionado. * Campos requeridos: Título, *n_resource_type, n_resource_group*.		
Bloqueada por	TFG-45	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 24 - Requisito TFG-46 – Funcionalidad Crear/Ver/Editar Recurso

Título	Ajustes/Mi perfil	Tipo	Épica
Id ref.	TFG-56	Épica	-
Descripción	-		
Bloqueada por	TFG-12	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	-

Tabla 25 - Requisito TFG-56 – Ajustes/Mi perfil

Título	Diseño Mi perfil	Tipo	HU
Id ref.	TFG-57	Épica	Ajustes/Mi perfil
Descripción	* Generar vista de diseño de Mi Perfil. Incluye la foto del usuario, nombre, apellidos y mail. * Botones de Editar perfil, notificaciones, acerca de, ayuda y logout.		
Bloqueada por	-	Bloqueante de	TFG-58
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 26 - Requisito TFG-57 – Diseño Mi perfil

Título	Funcionalidad Mi perfil	Tipo	HU
Id ref.	TFG-58	Épica	Ajustes/Mi perfil
Descripción	* Muestra el listado de botones. Pulsando encima de cada ítem nos lleva a la vista seleccionada.		
Bloqueada por	TFG-57	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 27 - Requisito TFG-58 – Funcionalidad Mi perfil

Título	Diseño Editar Perfil	Tipo	HU
Id ref.	TFG-59	Épica	Ajustes/Mi perfil
Descripción	* Muestra foto de perfil. * Muestra formulario con los datos del usuario. * Botón de Cancelar y Aceptar. * Botón de volver atrás arriba a la izquierda.		
Bloqueada por	-	Bloqueante de	TFG-60
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 28 - Requisito TFG-59 – Diseño Editar Perfil

Título	Funcionalidad Editar Perfil	Tipo	HU
Id ref.	TFG-60	Épica	Ajustes/Mi perfil
Descripción	* Permite editar todos los campos del registro. * Cambiar contraseña. * Pulsando cancelar o atrás vuelve a la vista de Mi perfil. * Pulsando a Aceptar realiza llamada a la API para actualizar los datos del usuario. * Si algún campo está mal introducido no permite realizar la opción de Aceptar. * Pulsando en la imagen del usuario permite modificarla por una existente en la galería del usuario o tomar una nueva de la cámara.		
Bloqueada por	TFG-59	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 29 - Requisito TFG-60 – Funcionalidad Editar Perfil

Título	Diseño Configuración Notificaciones	Tipo	HU
Id ref.	TFG-61	Épica	Ajustes/Mi perfil
Descripción	* Muestra dos seleccionables para push y correo. * Botón atrás arriba a la izquierda.		
Bloqueada por	-	Bloqueante de	TFG-62
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 30 - Requisito TFG-61 – Diseño Configuración Notificaciones

Título	Diseño Acerca de	Tipo	HU
Id ref.	TFG-64	Épica	Ajustes/Mi perfil
Descripción	* Muestra el logo de la empresa, nombre de la empresa, versión de la app y una breve descripción. * Botón de atrás arriba a la izquierda.		
Bloqueada por	-	Bloqueante de	TFG-65
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 31 - Requisito TFG-64 – Diseño Acerca de

Título	Funcionalidad Acerca de	Tipo	HU
Id ref.	TFG-65	Épica	Ajustes/Mi perfil
Descripción	* Pulsando el botón de atrás nos lleva a la vista de Mi perfil.		
Bloqueada por	TFG-64	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	10/03/2016

Tabla 32 - Requisito TFG-65 – Funcionalidad Acerca de

3.2.3 Sprint 3

Título	Mensajes	Tipo	Épica
Id ref.	TFG-29	Épica	-
Descripción	-		
Bloqueada por	TFG-12	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	-

Tabla 33 - Requisito TFG-29 – Mensajes

Título	Diseño Mensajes Android	Tipo	HU
Id ref.	TFG-30	Épica	Mensajes
Descripción	* *Select* en la parte superior (mensajes enviados o recibidos). * Botón de lupa para buscar arriba a la derecha. * Listado de mensajes que incluye la inicial del emisor del mensaje, título del mensaje y fecha. * Botón de añadir mensaje con el botón FAB de *Material Design*.		
Bloqueada por	-	Bloqueante de	TFG-31
Fecha creación	05/02/2016	Fecha realización	04/04/2016

Tabla 34 - Requisito TFG-30 – Diseño Mensajes Android

Título	Funcionalidad Mensajes	Tipo	HU
Id ref.	TFG-31	Épica	Mensajes
Descripción	* Si pulsamos el botón de enviar mensaje abre la vista "Mostrar/Redactar Mensaje" con los campos vacíos. * La barra de búsqueda filtra los mensajes con la palabra clave que hayamos introducido. * Si pulsamos una celda, muestra la vista "Mostrar/Redactar Mensaje" con los datos del mensaje seleccionado. * Dependiendo de la bandeja seleccionada nos actualiza la tabla con los mensajes que tengan el usuario en *From* o en *To*.		
Bloqueada por	TFG-30, TFG-89	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	04/04/2016

Tabla 35 - Requisito TFG-31 – Funcionalidad Mensajes

Título	Diseño Vista Mensaje	Tipo	HU
Id ref.	TFG-32	Épica	Mensajes
Descripción	* Botón de atrás arriba a la izquierda. * Botón de eliminar y responder arriba a la derecha. * Título del mensaje. * *From*. * Cuerpo del mensaje.		
Bloqueada por	-	Bloqueante de	TFG-33
Fecha creación	05/02/2016	Fecha realización	04/04/2016

Tabla 36 - Requisito TFG-32 – Diseño Vista Mensaje

Título	Funcionalidad Vista Mensaje	Tipo	HU
Id ref.	TFG-33	Épica	Mensajes
Descripción	* Pulsando atrás vuelve a la *home* de Mensajes. * Pulsando en eliminar nos muestra *popup* de confirmación. * Pulsando en responder nos muestra la vista de Responder mensaje.		
Bloqueada por	TFG-32	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	04/04/2016

Tabla 37 - Requisito TFG-33 – Funcionalidad Vista Mensaje

Título	Diseño Nuevo/Responder Mensaje	Tipo	HU
Id ref.	TFG-34	Épica	Mensajes
Descripción	* Campo *to*: vacío en nuevo mensaje, relleno en responder mensaje. * Título: vacío en nuevo, RE:<título mensaje anterior> en responder mensaje. * Cuerpo de mensaje. * Botón de enviar.		
Bloqueada por	-	Bloqueante de	TFG-35
Fecha creación	05/02/2016	Fecha realización	04/04/2016

Tabla 38 - Requisito TFG-34 – Diseño Nuevo/Responder Mensaje

Título	Funcionalidad Nuevo/Responder Mensaje	Tipo	HU
Id ref.	TFG-35	Épica	Mensajes
Descripción	* Campos "*to*" obligatorios. * Pulsando en enviar mensaje se enviará el mensaje.		
Bloqueada por	TFG-34	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	04/04/2016

Tabla 39 - Requisito TFG-35 – Funcionalidad Nuevo/Responder Mensaje

Título	Modificación petición mensajes Backend	Tipo	HU
Id ref.	TFG-89	Épica	Mensajes
Descripción	* Modificar la petición de devolución de mensajes para que devuelva los mensajes donde el usuario es el emisor.		
Bloqueada por	-	Bloqueante de	TFG-31
Fecha creación	05/02/2016	Fecha realización	04/04/2016

Tabla 40 - Requisito TFG-89 – Modificación petición mensajes Backend

3.2.4 Sprint 4

Título	Contactos	Tipo	Épica
Id ref.	TFG-51	Épica	-
Descripción	-		
Bloqueada por	TFG-12	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	-

Tabla 41 - Requisito TFG-51 – Contactos

Título	Diseño Contactos	Tipo	HU
Id ref.	TFG-52	Épica	Contactos
Descripción	* Muestra el listado con todas las empresas de la base de datos.		
Bloqueada por	-	Bloqueante de	TFG-53
Fecha creación	05/02/2016	Fecha realización	18/04/2016

Tabla 42 - Requisito TFG-52 – Diseño Contactos

Título	Funcionalidad Contactos	Tipo	HU
Id ref.	TFG-53	Épica	Contactos
Descripción	* Pulsando sobre la empresa abre el listado de contactos. * Pulsando el botón de navegación abre *popup* con las opciones de medio de transporte e inicia la navegación. * Pulsando el botón de mapa abre *popup* con los navegadores que tenga el usuario por defecto instalados en el dispositivo. Se pasan por parámetro la longitud y latitud de la empresa seleccionada. * Pulsando sobre el contacto abre vista con el detalle del contacto. * Pulsando sobre el email de contacto abre el cliente de correo que tenga en el dispositivo para enviar nuevo correo. * Pulsando sobre el teléfono de contacto abre el terminal para realizar la llamada.		
Bloqueada por	TFG-52	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	18/04/2016

Tabla 43 - Requisito TFG-53 – Funcionalidad Contactos

Título	*Material Design Calendar*	Tipo	HU
Id ref.	TFG-93	Épica	Registro de Horas
Descripción	* Integrar *Material Design Calendar* para Android e importar la librería.		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	18/04/2016

Tabla 44 - Requisito TFG-93 – Material Design Calendar

Título	Recursos y permisos	Tipo	HU
Id ref.	TFG-96	Épica	Recursos
Descripción	* Los recursos del usuario deberán aparecer en función de los permisos dados por el administrador de la Intranet.		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	18/04/2016

Tabla 45 - Requisito TFG-96 – Recursos y permisos

Título	Traducir la app	Tipo	HU
Id ref.	TFG-97	Épica	-
Descripción	* Generar los .xml necesarios para la traducción a varios idiomas inglés y español (por defecto)		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	18/04/2016

Tabla 46 - Requisito TFG-97 – Traducir app

Título	Cambiar foto de perfil	Tipo	HU
Id ref.	TFG-106	Épica	Ajustes/Mi perfil
Descripción	* Cambiar petición POST para almacenar imagen en la intranet		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	13/05/2016

Tabla 47 - Requisito TFG-106 – Cambiar foto de perfil

3.2.5 Sprint 5

Título	Funcionalidad Configuración Notificaciones	Tipo	HU
Id ref.	TFG-62	Épica	Ajustes/Mi perfil
Descripción	* Pulsando en el seleccionable se activa/desactiva las notificaciones *push* o correo. Realiza la llamada a la API para modificar las notificaciones en la sesión del usuario. * Pulsando atrás vuelve a la vista Mi Perfil.		
Bloqueada por	TFG-61, TFG-63, TFG-90	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	13/05/2016

Tabla 48 - Requisito TFG-62 – Funcionalidad Configuración Notificaciones

Título	Notificaciones Backend	Tipo	HU
Id ref.	TFG-63	Épica	Ajustes/Mi perfil
Descripción	* Gestión de notificaciones para el usuario seleccionado en la base de datos (contemplar si se requiere notificación por push, correo, ambas o ninguna).		
Bloqueada por	-	Bloqueante de	TFG-62
Fecha creación	05/02/2016	Fecha realización	13/05/2016

Tabla 49 - Requisito TFG-63 – Notificaciones Backend

Título	Wearable	Tipo	Épica
Id ref.	TFG-83	Épica	-
Descripción	-		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	-

Tabla 50 - Requisito TFG-83 – Wearable

Título	Diseño de la vista del wearable	Tipo	HU
Id ref.	TFG-84	Épica	Wearable
Descripción	* Notificación de nuevo mensaje. * *Sweep* a la izquierda para ver el detalle del mensaje, responder mensaje con texto predefinido y comando de voz, abrir app en el dispositivo y cancelar notificaciones.		
Bloqueada por	-	Bloqueante de	TFG-85
Fecha creación	05/02/2016	Fecha realización	13/05/2016

Tabla 51 - Requisito TFG-84 – Diseño de la vista del wearable

Título	Funcionalidad wearable	Tipo	HU
Id ref.	TFG-85	Épica	Wearable
Descripción	* Pulsando en el mensaje posibilidad de ver el cuerpo del mensaje.		
Bloqueada por	TFG-84	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	13/05/2016

Tabla 52 - Requisito TFG-85 – Funcionalidad wearable

Título	Notificaciones Wearable Backend	Tipo	HU
Id ref.	TFG-90	Épica	Wearable
Descripción	* Realizar modificación en el backend para establecer conexión con GCM.		
Bloqueada por	-	Bloqueante de	TFG-62
Fecha creación	05/02/2016	Fecha realización	13/05/2016

Tabla 53 - Requisito TFG-90 – Notificaciones Wearable Backend

Título	Cambiar peticiones a POST	Tipo	HU
Id ref.	TFG-110	Épica	-
Descripción	* Cambiar la clase Rest de la aplicación móvil para realizar las peticiones por POST.		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	19/04/2016

Tabla 54 - Requisito TFG-110 – Cambiar peticiones a POST

3.2.6 Sprint 6

Título	Memorias	Tipo	Épica
Id ref.	TFG-111	Épica	-
Descripción	-		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	-

Tabla 55 - Requisito TFG-111 - Memorias

Título	Memoria Android	Tipo	HU
Id ref.	TFG-112	Épica	Memorias
Descripción	* Redacción memoria Android.		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	

Tabla 56 - Requisito TFG-112 – Memoria Android

Título	Revisión App	Tipo	Épica
Id ref.	TFG-114	Épica	-
Descripción	-		
Bloqueada por	TFG-12	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	-

Tabla 57 - Requisito TFG-114 – Revisión App

Título	Bugs Android	Tipo	HU
Id ref.	TFG-115	Épica	Revisión App
Descripción	* Implementar soluciones después del testeo final de la aplicación.		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	

Tabla 58 - Requisito TFG-115 – Bugs Android

Título	Diseño Android	Tipo	HU
Id ref.	TFG-117	Épica	Revisión App
Descripción	* Modificar el diseño y los iconos de la aplicación si fuera necesario.		
Bloqueada por	-	Bloqueante de	-
Fecha creación	05/02/2016	Fecha realización	

Tabla 59 - Requisito TFG-117 – Diseño Android

A continuación podemos ver el diagrama de avance del proyecto a lo largo de los meses de desarrollo según la metodología Scrum.

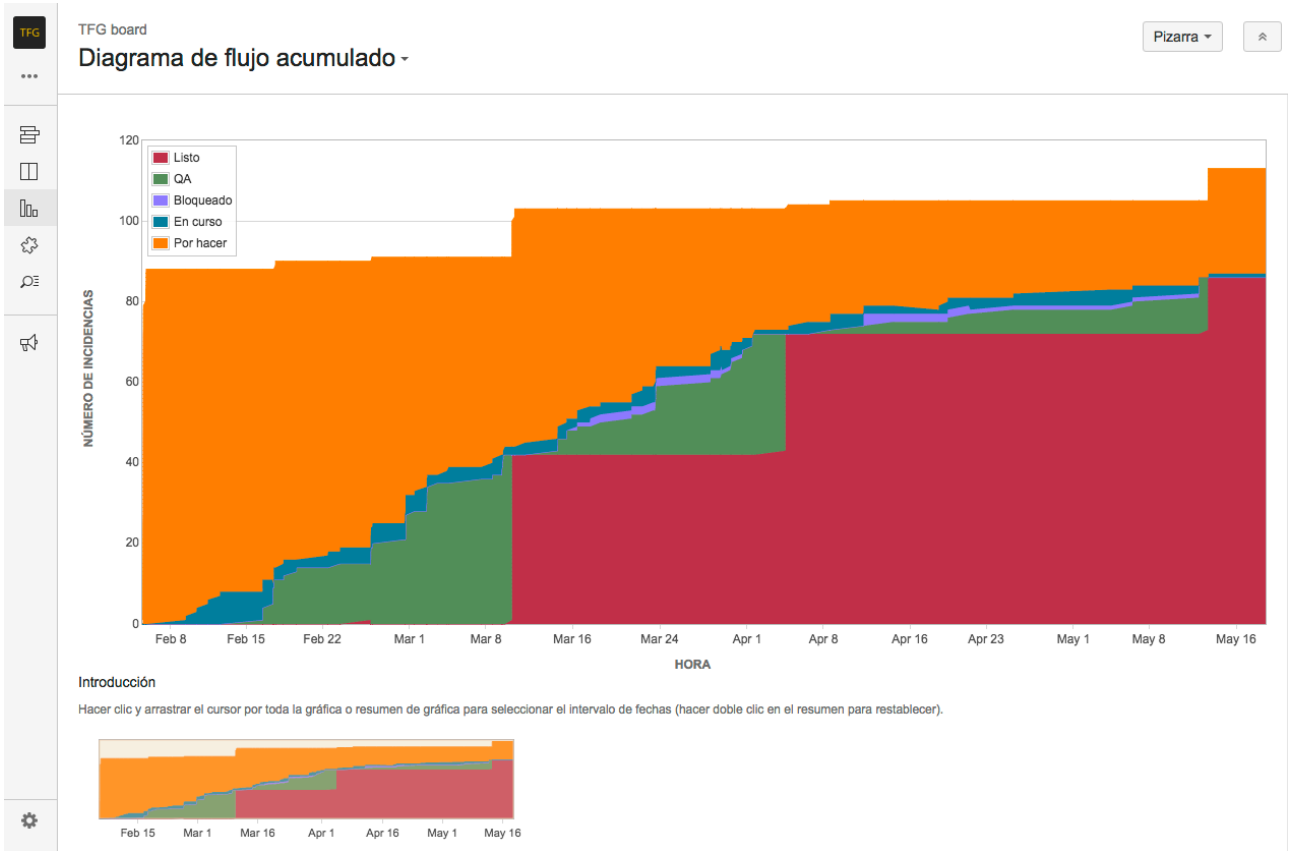

Ilustración 15 – Diagrama de flujo acumulado de la aplicación.

3.2.7 Servicios implementados de la API

Para que la aplicación móvil pueda obtener la información necesaria en cada módulo ha sido necesario la implementación de diversos servicios en la API. En este apartado se detallarán todos los servicios añadidos y se explicarán con mayor detalle aquellos que he tenido que realizar.

Todas las peticiones realizadas a la API siguen la siguiente estructura con los parámetros de entrada:

- **service**: nombre del servicio al que se quiere realizar la petición.
- **auth**: *token de autorización* que se devuelve al usuario cuando inicia sesión. Este se almacena en el dispositivo móvil y se incluye en todas las peticiones (menos en el login) para dotar a la aplicación de seguridad ante ataques externos. Este *token* sirve para que no se pueda acceder a la información de la base de datos si no se está autorizado. Cada *token* va ligado a un identificador de usuario, de esta manera, el usuario solo podrá acceder a su propia información.
- **args**: argumentos de entrada de la petición. Estos difieren dependiendo de los parámetros de entrada que necesite cada servicio.

Los servicios implementados en la API se dividen en módulos, tal y como lo hace la aplicación con el fin de tener una estructura ordenada. Las tablas que se muestran a continuación se componen de:

- **Servicio (Módulo)**: nombre del servicio al cual se realiza la petición y entre paréntesis el módulo de la API al que pertenece.
- **Parámetros de entrada**: JSON que compone la variable **args** detallada en el párrafo anterior.
- **Descripción**: breve descripción de la funcionalidad del servicio.

El proyecto se ha realizado paralelamente con otro compañero (tecnología iOS), por lo que decidimos repartirnos las tareas del desarrollo de la API del servicio web para dar soporte a las aplicaciones móviles, tanto Android como iOS. A continuación se listarán los servicios desarrollados por mi compañero de equipo:

Servicio (Módulo)	Parámetros de entrada	Descripción
deleteResource (ResourceService)	**id**: identificador del recurso a eliminar	Elimina de la base de datos el recurso que se le pasa por parámetro.
addResource (ResourceService)	**title**: título del recurso. **url**: URL del recurso. **username**: nombre de usuario. **password**: contraseña. **comment**: comentario. **fk_n_resource_type**: identificador de tipo de recurso. **fk_n_resource_group**: identificador de grupo de recurso.	Añade a la base de datos un nuevo recurso con la información pasada por parámetro en la petición.
updateResource (ResourceService)	**id**: identificador del recurso. **title**: nuevo título del recurso. **url**: nueva URL del recurso. **username**: nuevo nombre de usuario. **password**: nueva contraseña. **comment**: nuevo comentario. **fk_n_resource_type**: identificador de tipo de recurso. **fk_n_resource_group**: identificador de grupo de recurso.	Actualiza la base de datos con la nueva información correspondiente al recurso pasado por parámetro.

getResourcesForGroup (ResourceService)	**group**: identificador de grupo de recursos.	Devuelve el listado de recursos correspondiente al grupo de recursos que se pasa por parámetro.
getMyMessages (MailService)	-	Devuelve los mensajes recibidos por el usuario. El usuario se obtiene por medio del token de autentificación (auth).
getSentMessages (MailService)	-	Devuelve los mensajes enviados por el usuario.
sendMessage (MailService)	**to**: identificador del usuario de destino. **subject**: título del mensaje. **message**: cuerpo del mensaje. **push_notification_message**: mensaje que se mostrará en la notificación (por defecto, "Nuevo mensaje de" + nombre del remitente).	Guarda en la base de datos el nuevo mensaje del usuario relacionándolo con el emisor del mensaje y el receptor. Comprueba si el receptor del mensaje tiene las notificaciones push activas, si es así, comprueba si es Android o iOS y llama al método sendAndroidNotification o sendIOSNotification respectivamente.
deleteMessage (MailService)	**message_id**: identificador del mensaje. **option**: 0 ó 1, dependiendo si el usuario que elimina el mensaje es el emisor o el receptor.	Desvincula en la base de datos el mensaje a eliminar con el identificador de usuario. De esta manera no se le mostrará al usuario que lo ha eliminado pero seguirá mostrándose al que no lo ha borrado.
setMailRead (MailService)	**id**: identificador del mensaje.	Marca como leído el mensaje por el receptor cuando lo abre en la aplicación (poniendo el campo "read" a true en la base de datos).
sendIOSNotification (NotificationsService)	**device_id**: token asignado por APN (Apple Push Notification) del dispostivo destinatario. **message**: datos del mensaje.	Servicio llamado al enviar un mensaje por el usuario que se comunica con APN

	notification_type: 1 ó 2, dependiendo si es una notificación de aviso o de mensaje.	para enviar la notificación push al destinatario.
getResource_groupForUser (Resource_groupService)	-	Devuelve los grupos de recurso disponibles para el usuario que realiza la petición (el acceso a los grupos de recursos es gestionado por el administrador de la intranet).
getCompanies (CompanyService)	-	Devuelve el listado de compañías clientes de la empresa.

Tabla 60 - Definición servicios implementados por el equipo de desarrollo.

La extensión de los servicios puede ser larga para este documento, por tanto se expondrá únicamente la composición de dos de ellos como ejemplo, en concreto, *sendAndroidNotification* para explicar la comunicación con GCM y *updateProfileImage* para explicar la transmisión y gestión de la imagen. A continuación se mostrará la tabla con los servicios desarrollados por mí siguiendo la misma estructura que la tabla anterior.

Servicio (Módulo)	Parámetros de entrada	Descripción
getTimeTrackForDate (Time_trackService)	**date**: fecha en formato *aaaa-mm-dd*.	Devuelve el listado de horas imputadas por el usuario para un día en concreto.
updateTimeTrack (Time_trackService)	**hours**: número de horas. **fk_n_time_track_type**: tipo de imputación. **description**: breve descripción del trabajo realizado. **fk_n_resource_group**: nombre del proyecto al que se le asignan las horas.	Actualiza la base de datos con la información que se le pasa como parámetro.
deleteTimeTrack (Time_trackService)	**id**: identificador único de la hora a eliminar.	Elimina de la base de datos la imputación de la hora que se le pasa como parámetro.
addTimeTrack (Time_trackService)	**hours**: número de horas.	Crea una nueva imputación en la base

	fk_n_time_track_type: tipo de imputación. **description**: breve descripción del trabajo realizado. **fk_n_resource_group**: nombre del proyecto al que se le asignan las horas.	de datos para un día concreto relacionándola con el identificador de usuario.
getTime_track_type (Time_track_typeService)	-	Devuelve los tipos de imputación disponibles en la base de datos.
sendAndroidNotification (NotificationsService)	**device_id**: token asignado por GCM del dispostivo destinatario. **message**: texto de la notificación. **detail**: datos del mensaje (título y cuerpo del mensaje). **detail_user**: detalle del emisor del mensaje. **notification_type**: 1 ó 2, dependiendo si es una notificación de aviso o de mensaje.	Servicio llamado al enviar un mensaje por el usuario (*sendMessage*) que se comunica con GCM para enviar la notificación push al destinatario.
updateUser (UsersService)	**firstname**: nombre del usuario. **lastaname**: apellido del usuario. **email**: email del usuario. **nif**: NIF del usuario.	Actualiza la información básica del usuario de la base de datos.
updatePassword (UsersService)	**newPassword**: nueva contraseña.	Modifica la contraseña actual del usuario.
getUsers (UsersService)	-	Devuelve todos los usuarios de la empresa.
updateDeviceId (UsersService)	**device_id**: token generado por GCM que se le asigna al usuario. **OS**: sistema operativo (Android o iOS)	Actualiza en la base de datos el token del usuario asignado por GCM/APN si éste ha cambiado.
updateProfileImage (UsersService)	**tmp_name**: nombre del usuario que ha realizado la foto. Se le incluye al final un número aleatorio para no sobrescribir la imagen de otro usuario.	Actualiza la foto de perfil del usuario. Crea un nuevo fichero en el servidor y lo referencia

	photo: bytes de la foto codificados con *base64*.	al identificador del usuario.
getPanelInfo (MainService)	-	Devuelve los 3 últimos registros (mensajes, horas imputadas y recursos) del usuario.
getSalesContactForcompany (Sales_contactService)	**fk_n_company**: identificador de la compañía.	Devuelve los contactos pertenecientes a la compañía que se le pasa como parámetro.

Tabla 61 - Definición servicios implementados por mi.

Todos los servicios tienen como parámetro de entrada el *user_id*, obtenido con el *auth* que se incluye en cada petición (ya que el token de autorización es único para cada usuario).

Los servicios de la API han sido implementados en tecnología PHP. A continuación explicaremos como se ha implementado el servicio de *sendAndroidNotification* y *updateProfileImage*:

sendAndroidNotification

Para enviar notificaciones push a través de GCM debemos realizar una petición POST a la siguiente URL:

```
$url = 'https://android.googleapis.com/gcm/send';
```

En la cabecera de la petición debermos indicar el proyecto relacionado y que es tipo JSON:

```
$headers = array(
    'Authorization: key=' . GOOGLE_API_KEY,
    'Content-Type: application/json'
);
```

Tras esto, le indicaremos los campos que llevará la petición. En nuestro caso, el dispositivo de destino y los datos que llevará la notificación:

```
$fields = array(
    'registration_ids' => [$device_id],
    'data' => $data,
);
```

Por último realizaremos la petición[11] mediante *curl_init()* para iniciar sesión, *curl_setopt()* para establecer las opciones, *curl_exec()* para ejecutar la petición y *curl_close()* para cerrar la sesión:

```
   $ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_POST, count($fields));
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
curl_setopt($ch, CURLOPT_POSTFIELDS, json_encode($fields));
$result = curl_exec($ch);
if ($result === FALSE) {
    die('Curl failed: ' . curl_error($ch));
}
curl_close($ch);
```

updateProfileImage

Desde la aplicación se obtiene el mapa de bits (bitmap) de la imagen capturada con la cámara o la imagen de la galería y se procede a codificar la imagen con Base64 y convertirlo a String de la siguiente manera:

```
String imgString = Base64.encodeToString(getBytesFromBitmap(bitmap),
      Base64.NO_WRAP);
```

Es esta variable lo que se pasa como parámetro en la petición al servicio. En el lado del servidor procederemos a decodificar la imagen y generar un nuevo fichero con el nombre del usuario seguido de números aleatorios con el objetivo de no sobrescribir ningún archivo:

```
   $data = base64_decode($args['photo']); //decodificación del archivo
$filename = $args['tmp_name'] . '_' . rand(0, 9999999); //generación del
nombre
$sql = "UPDATE k_users SET url_pic = '$filename' WHERE id = '$id'";
$this->bbdd->sendQuery($sql);
$target_path = $_SERVER['DOCUMENT_ROOT'] . "/intranet/uploads/" .
$filename . ".jpg"; //asignación de la carpeta de destino dentro del
servidor
if (file_put_contents($target_path, $data)) { //generación del archivo
    $mRet = array("url_pic" => "$filename");
} else {
    $mRet = array("command" => "error");
}
```

3.2.8 Restricciones de diseño y funcionalidad

La aplicación ha sido diseñada y optimizada para la versión **4.4** de Android denominada Kitkat (API 19). La mínima versión para poder ejecutar la aplicación es la 4.0.3 denominada IceCream (API 15) y la máxima es la 6.0 denominada Marshmallow (API 23). De esta manera abracamos la amplia mayoría del mercado (97.7%). Todo el diseño se ha basado en la guía de diseño de Material Design de Android [16].

[11] php Documentation, *http://php.net/manual* [en línea]. Disponible en: http://php.net/manual/es/curl.examples-basic.php [Consulta: 17 de abril 2016]

En la aplicación se utilizan librerías propias de Android como externas, tanto para la parte de diseño como para dar funcionalidad. A continuación listamos las librerías utilizadas como dependencias en el *gradle* de la aplicación:

- v7 Support Library
  - Appcompat Library utilizada para dar soporte para ActionBar, AppCompatActivity, AppCompatDialog.
    - `'com.android.support:appcompat-v7:23.1.1'`
  - Cardview Library: soporte para el widget de CardView utilizado en Contactos.
    - `'com.android.support:cardview-v7:23.0.+'`
- v4 Support Library; s oporte para Fragments, notificaciones, Navigation Drawer.
  - `'com.android.support:support-v4:23.1.1'`
- Android Design Support Library: soporte para el diseño compatible de Material Design de TextInput, FABs, NavigationView.
  - `'com.android.support:design:23.1.1'`
- GCM: soporte para Google Cloud Messaging.
  - `"com.google.android.gms:play-services-gcm:8.4.0"`
- Librerías de terceros
  - amulyakhare/TextDrawable: soporte para imágenes con texto, utilizado en el módulo de mensajes de la aplicación.
    - `'com.amulyakhare:com.amulyakhare.textdrawable:1.0.1'`
  - prolificinteractive/material-calendarview: soporte para el calendario con Material Design, utilizado en el módulo de Imputación de horas de la aplicación.
    - `'com.prolificinteractive:material-calendarview:1.2.0'`
  - futuresimple/android-floating-action-button: soporte para dar funcionalidad a los Floating Action Buttons.
    - `'com.getbase:floatingactionbutton:1.10.1'`

Se incluyen 2 proyectos externos a lo que es la aplicación en sí:

- Volley [17]: librería para realizar peticiones http.
  - `project(':volley')`
- IntranetWear: proyecto que incluye la funcionalidad y configuración del wearable.
  - `project(':intranetwear')`

También se incluye un plugin:

- Google Services
  - `apply plugin: 'com.google.gms.google-services'`

En el proyecto de intranetwear incluimos varias dependencias:

- Wearable Support Library
  - `compile 'com.google.android.support:wearable:1.4.0'`
- Google Play Services
  - `compile 'com.google.android.gms:play-services-wearable:8.4.0'`

Información de la aplicación:

- Almacenamiento en el dispositivo: 6.88 MB.
- Uso RAM: 20 MB (estable) – 35 MB (realización de peticiones y carga de datos).

3.3 Descripción de la aplicación

A continuación se describirá la funcionalidad de los diferentes módulos de la aplicación móvil y wearable.

La aplicación tiene un menú lateral que contiene las vistas principales de la aplicación, en los que encontramos Panel Principal, Recursos, Registro de horas, Mensajes, Contactos y Ajustes/Mi perfil.

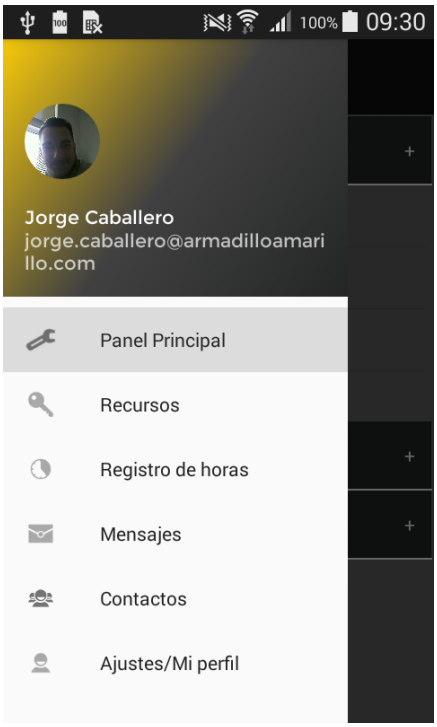

Ilustración 16 – Menú lateral – Diseño final de la aplicación.

Las imágenes que se muestran en esta sección se dividen en “Diseño inicial” (aquel diseño que se generó en una primera instancia cuando se creó la vista) y “Diseño final” (aquel diseño que finalmente se utiliza en la aplicación tras aplicar los cambios de estilo correspondientes).

En algunos módulos se incluye únicamente el diseño final de la aplicación porque los cambios entre el diseño inicial y éste son ínfimos o inexistentes.

3.3.1 Login

La primera pantalla que el usuario ve de la aplicación es la vista de **login**, en la cual tiene que introducir sus credenciales (aportadas por la empresa) para poder acceder a las funcionalidades de la aplicación.

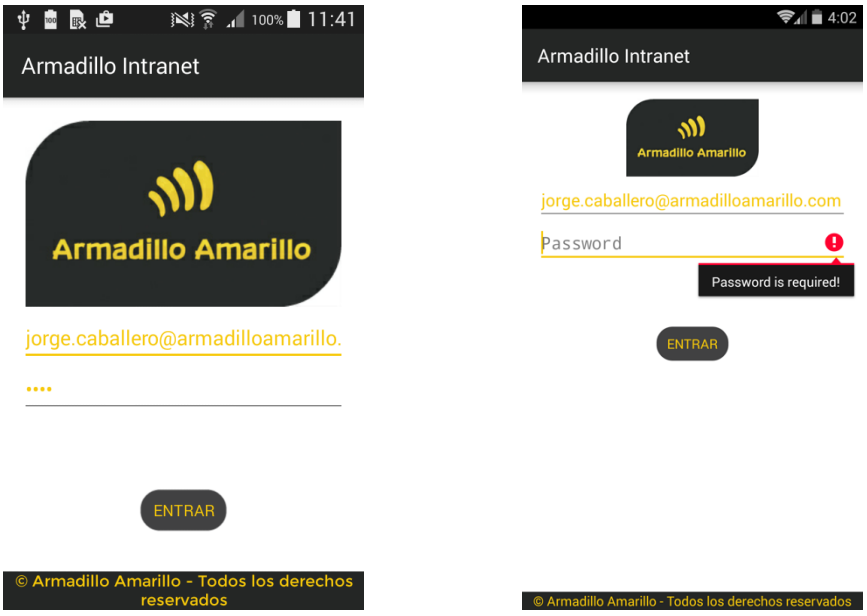

Ilustración 17 – Login – Diseño inicial de la aplicación.

Como podemos ver en la imagen anterior, es necesario introducir tanto *username* como *password* para poder enviar la información de iniciar sesión. También se incluye una opción para recordar las credenciales.

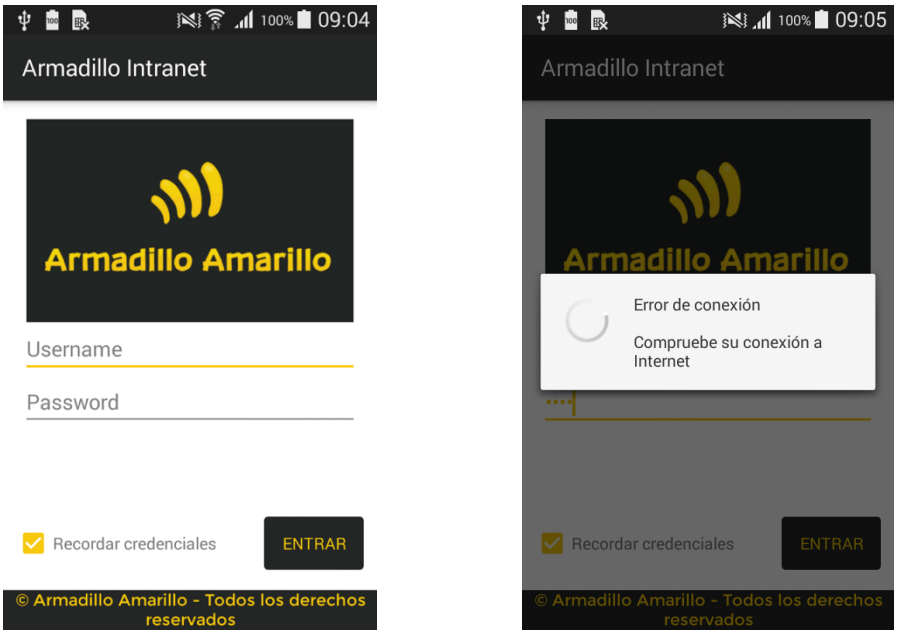

Ilustración 18 – Login – Diseño final de la aplicación.

3.3.2 Recursos

Esta vista muestra una lista con los **grupos de recurso** que están accesibles para el usuario. Se puede acceder a los **recursos específicos** de cada grupo pulsando encima de éste. También se muestra un botón para crear un nuevo recurso.

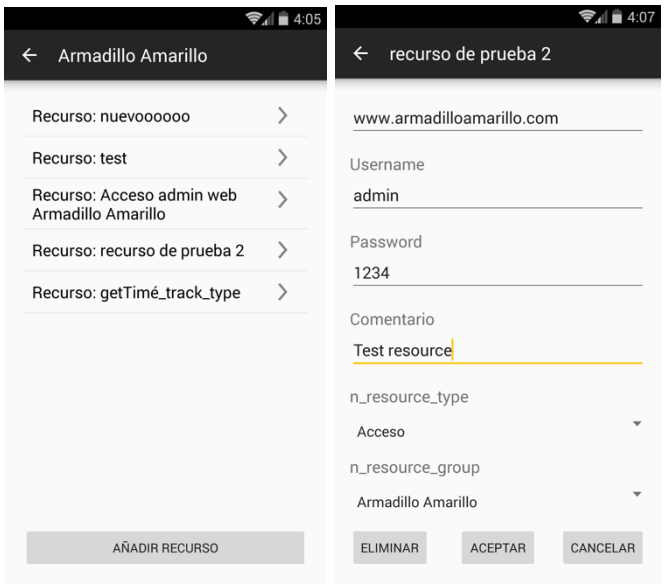

Ilustración 19 – Recursos –Diseño inicial de la aplicación.

Se puede acceder al detalle de cada recurso pulsando encima del mismo. De esta manera se abre una nueva vista que permite al usuario ver el recurso, editarlo y borrarlo.

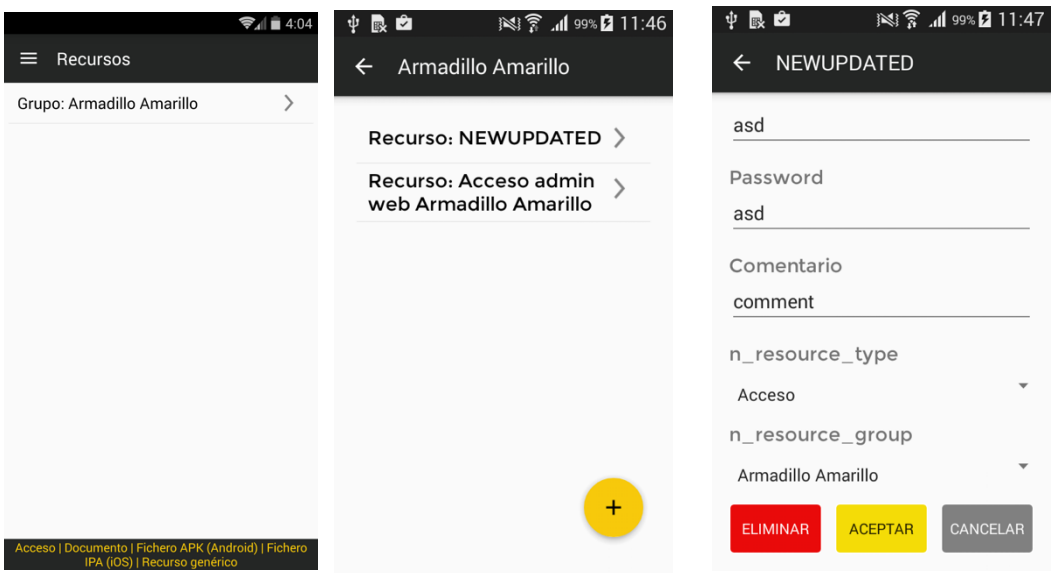

Ilustración 20 – Diseño final de la aplicación.

3.3.3 Registro de horas

Esta vista muestra un calendario en la parte superior de la pantalla del dispositivo centrándose en el día de hoy por defecto. En la parte inferior muestra un listado de las **horas imputadas** por el usuario en el día seleccionado. También mostrará un botón para imputar una nueva hora.

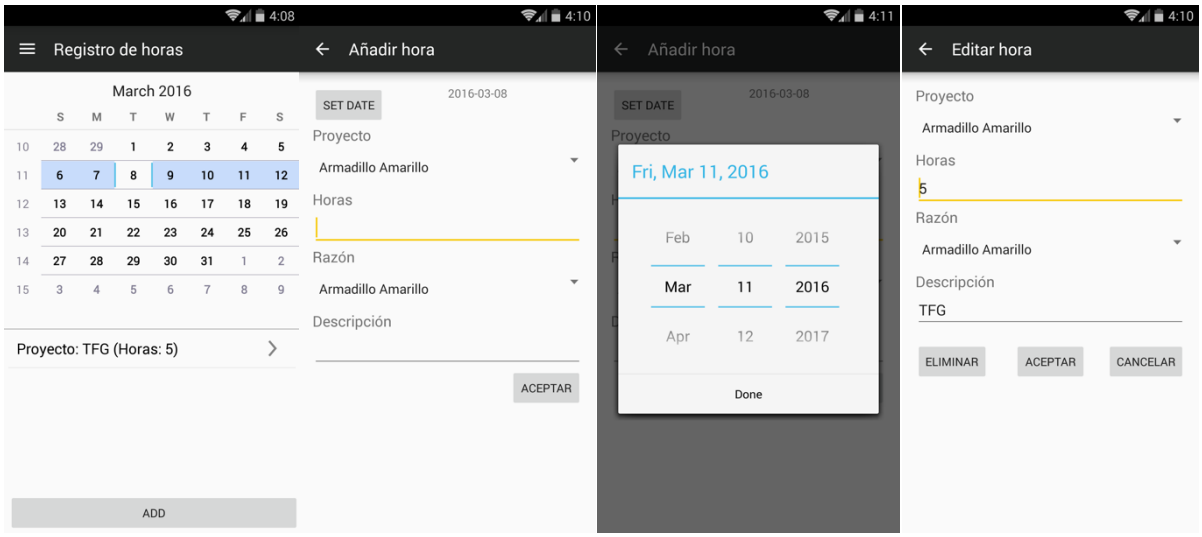

Ilustración 21 – Registro de horas –Diseño inicial de la aplicación.

Pulsando encima de cualquier registro de hora de la lista se accede al detalle del mismo, pudiendo editar o borrar.

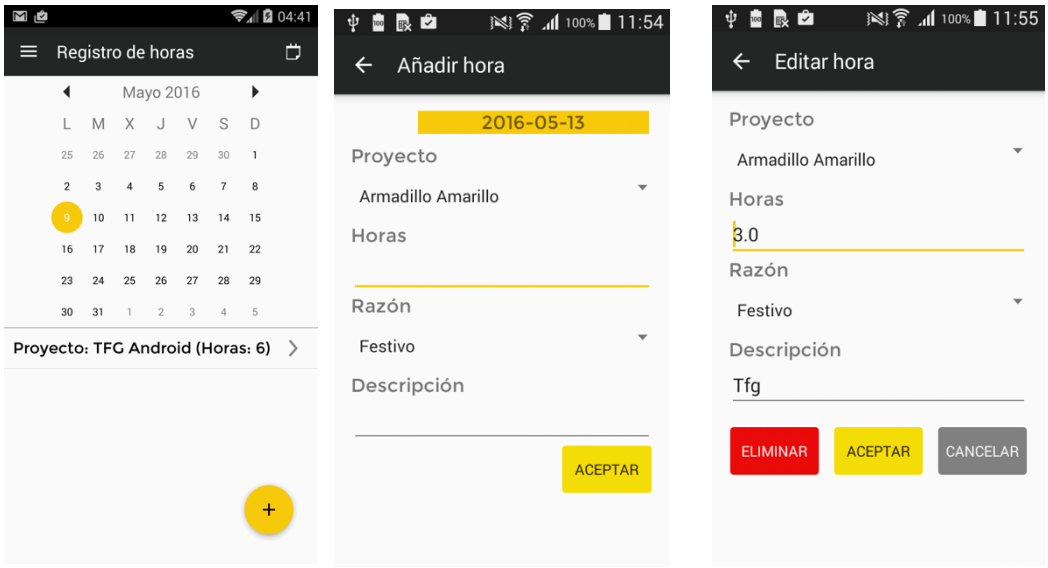

Ilustración 22 – Registro de horas –Diseño final de la aplicación.

3.3.4 Mensajes

Esta vista muestra la **bandeja de entrada y de salida** de los mensajes internos de la empresa. Cada bandeja muestra un listado con los diferentes mensajes del usuario.

En la bandeja de entrada se muestra el mensaje con diferente color de fondo dependiendo si este ha sido leído ya por el usuario (fondo gris) o si por el contrario todavía no ha sido abierto (fondo blanco).

Se ha implementado también una **barra de búsqueda** para que le resulte más fácil al usuario encontrar un mensaje en concreto. Esta búsqueda se filtra por *nombre de usuario, email, título de mensaje* y *cuerpo de mensaje*.

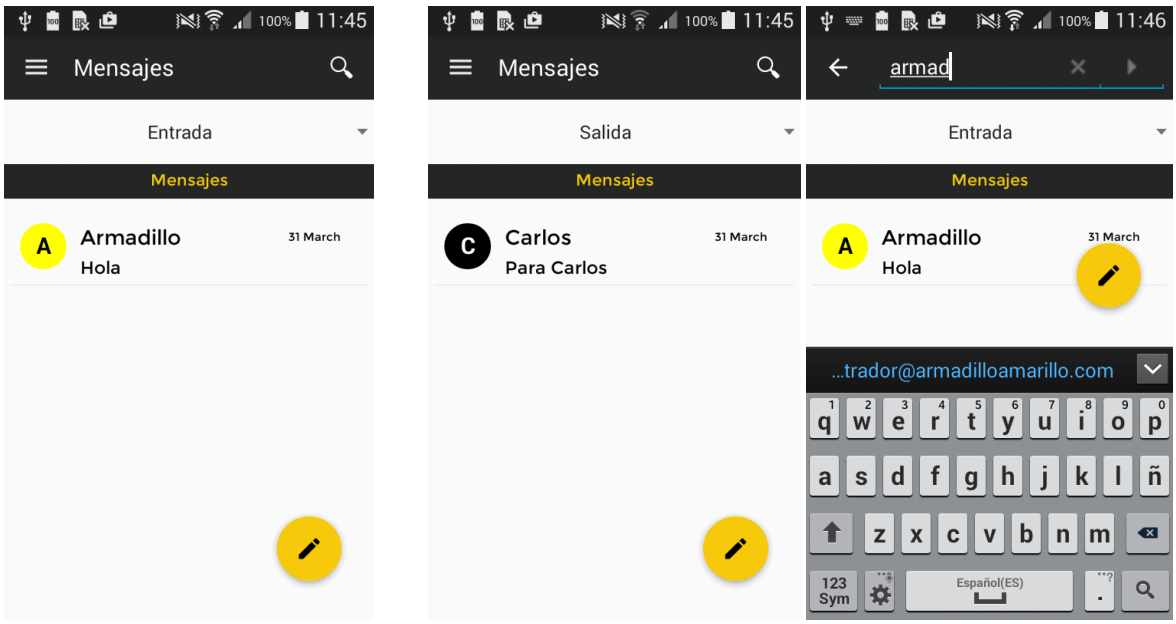

Ilustración 23 – Mensajes. Vista principal – Diseño final de la aplicación.

La vista principal también incluye un botón para generar un nuevo mensaje. Se abrirá una nueva vista compuesta por los registros básicos de un correo: *destinatario, asunto* y *cuerpo del mensaje*.

El destinatario es un campo de texto libre, pero al tratarse de mensajería interna de una empresa filtrará por el nombre y apellidos de los empleados de la misma, no siendo posible la comunicación con miembros externos a la compañía.

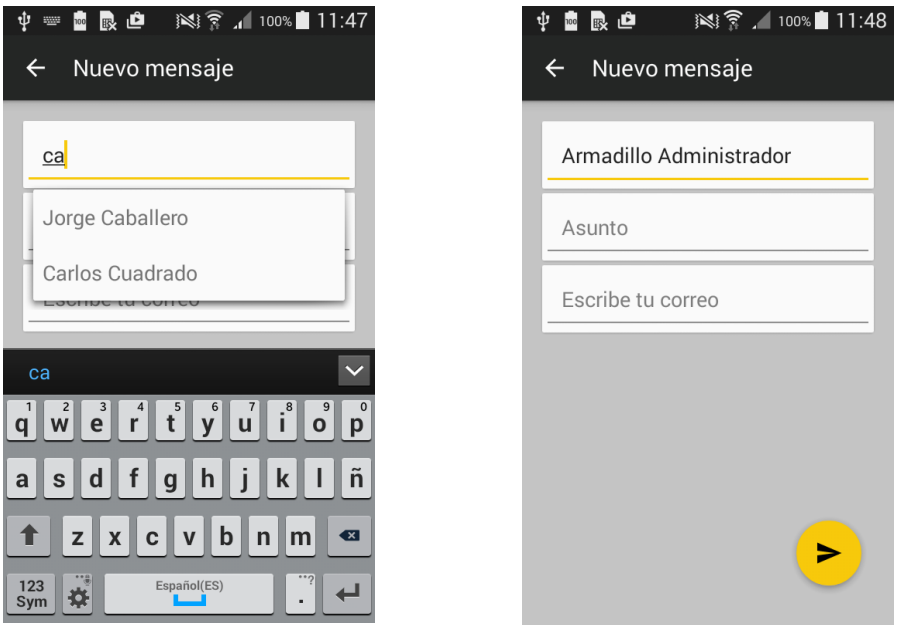

Ilustración 24 – Mensajes. Nuevo mensaje – Diseño final de la aplicación.

Pulsando encima de uno de los mensajes de la lista accedemos al detalle del mismo, pudiendo responder al mensaje o borrarlo de nuestra bandeja. Si el usuario pulsa para responder al mensaje se pondrá automáticamente el asunto del mensaje como *RE:<asunto>* y el texto citado del usuario al que se responde. Se incluye también un botón para enviar el mensaje.

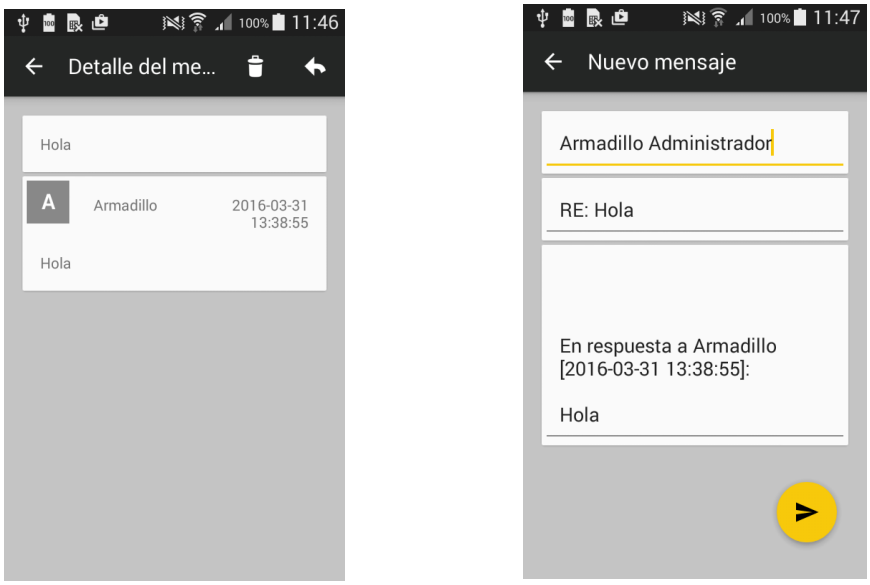

Ilustración 25 – Mensajes. Nuevo mensaje – Diseño final de la aplicación.

3.3.5 Contactos

Esta vista muestra un listado con las diferentes **empresas** clientes de la nuestra. Pulsando encima de cada una accedemos a los **contactos** directos que tenemos en los clientes con la posibilidad de acceder al detalle del mismo.

En el detalle del contacto se muestra:

- Información principal del contacto.
- Correo electrónico, con la posibilidad de crear un nuevo correo con dicho destinatario en el cliente de correo que se tenga instalado en el teléfono (Correo o Gmail).
- Teléfono fijo y móvil, con la posibilidad de iniciar una llamada telefónica (Se abrirá el marcador con el teléfono seleccionado ya introducido).

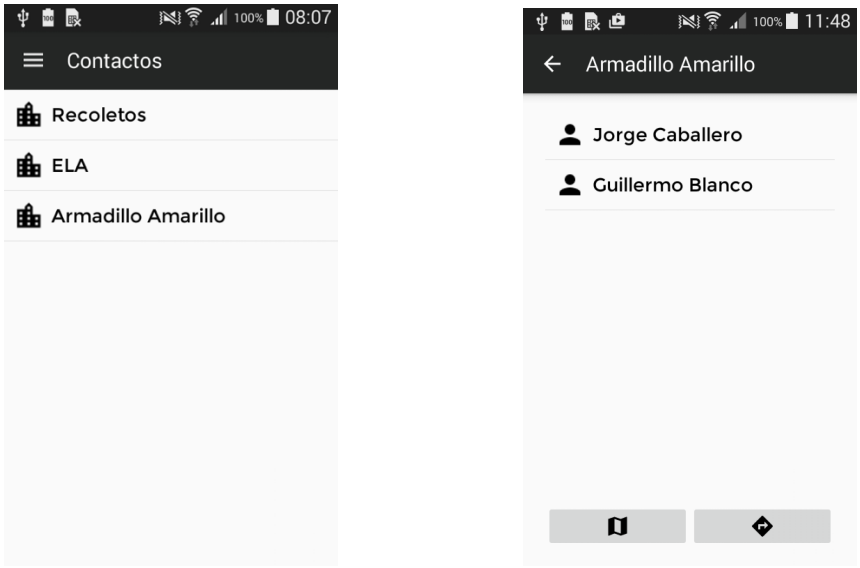

Ilustración 26 – Contactos. Detalle de la empresa y contacto –Diseño inicial de la aplicación.

En el detalle de la empresa se muestran dos botones contenidos en un botón flotante (FAB):

- Abrir la **localización** de la empresa seleccionada en *Google Maps*.
- Iniciar la **navegación** al destino a través de *Google Maps*. Se mostrará al usuario un *popup* para que indique el medio de transporte para iniciar la navegación.

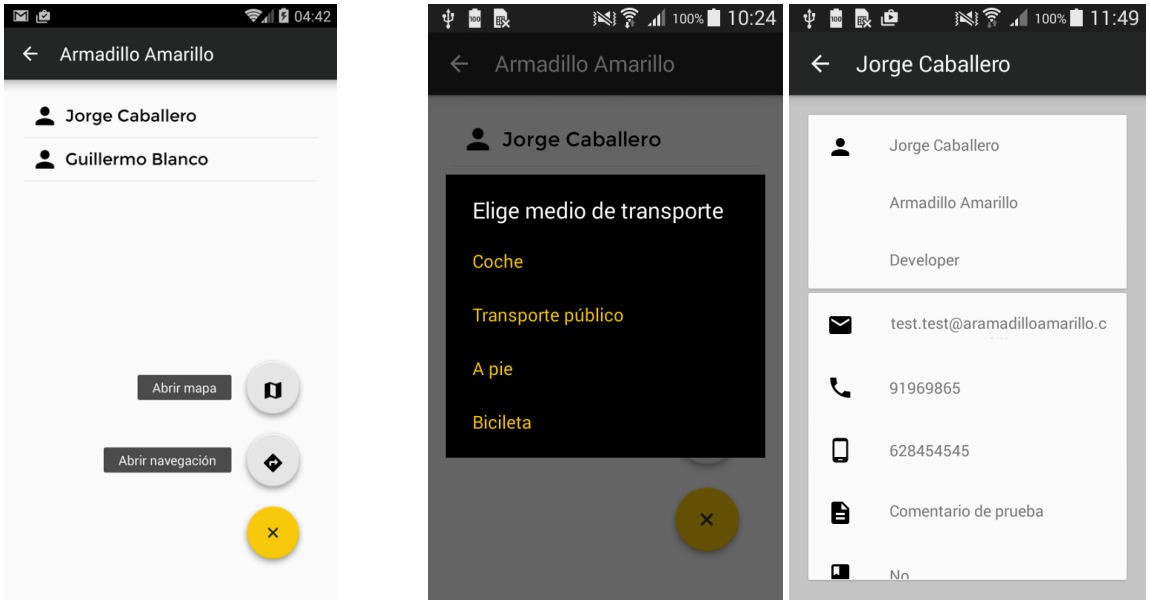

Ilustración 27 – Contactos. Detalle de la empresa –Diseño final de la aplicación.

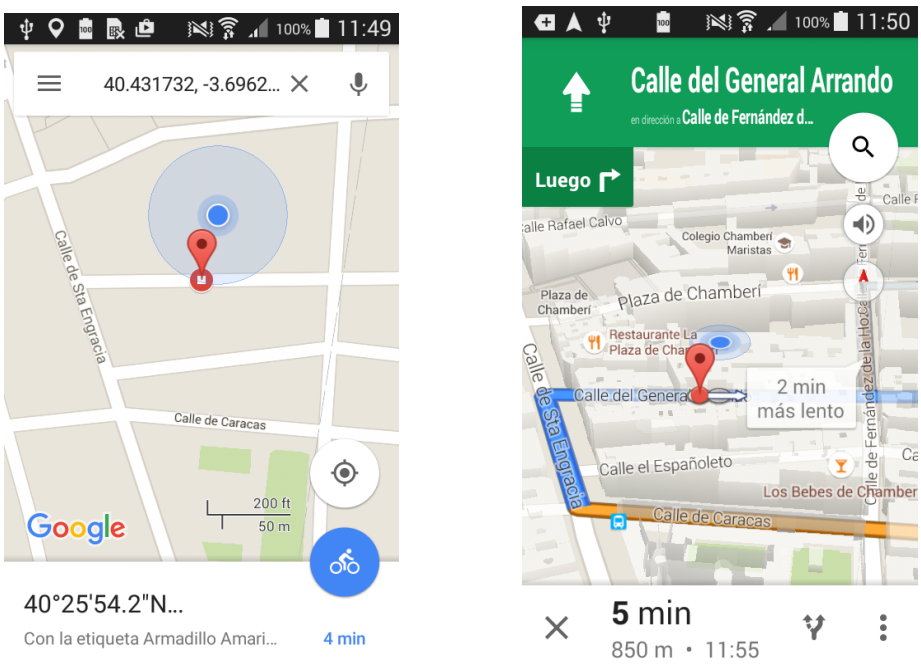

Ilustración 28 – Google Maps. Localización y navegación.

3.3.6 Perfil

Esta vista muestra las opciones para editar el perfil y los ajustes. Se incluye la opción de hacer logout de la aplicación. Se pueden ver:

- *Editar perfil*: permite editar los datos del usuario y cambiar la foto de perfil realizando una foto con la cámara u obteniendo una imagen de la galería del teléfono.
- *Cambiar contraseña*: permite cambiar la contraseña del usuario utilizada para iniciar sesión tanto en la aplicación móvil como en la web.
- *Configuración de notificaciones*: permite activar y desactivar las notificaciones push.
- *Ayuda*: facilita un correo electrónico al cual contactar por si tienes alguna duda sobre el uso de la aplicación.
- *Acerca de*: permite ver la versión de la aplicación que tienes instalada en el dispositivo y muestra un breve contexto de la empresa.
- *Logout*: cierra la sesión en la aplicación.

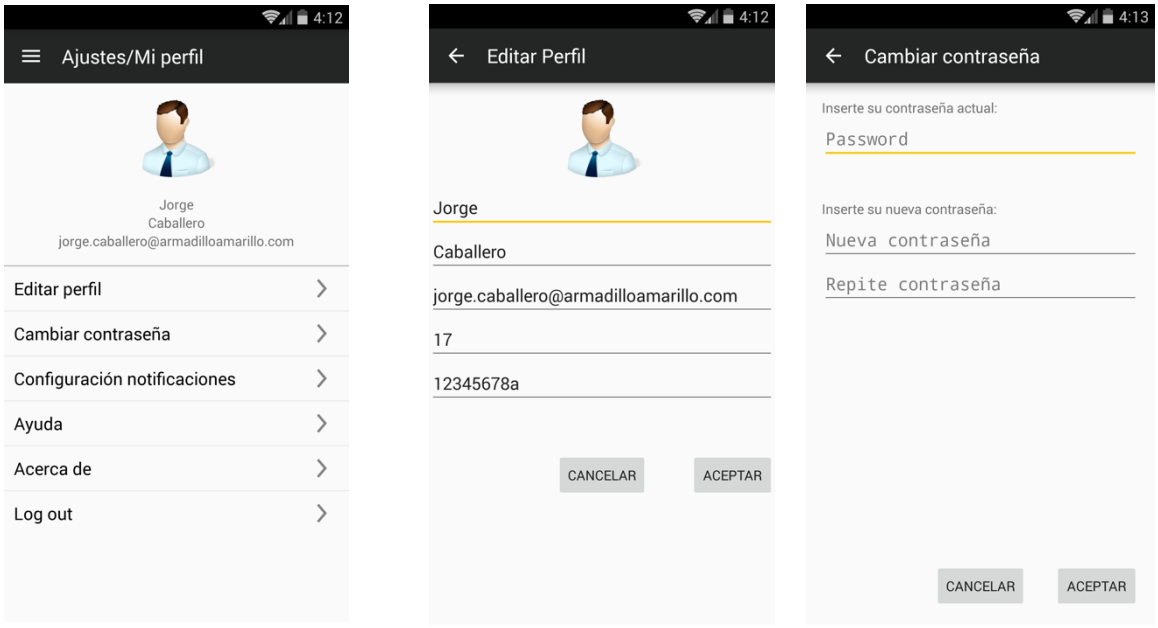

Ilustración 29 – Perfil. Vista principal, editar perfil y cambiar contraseña – Diseño inicial de la aplicación.

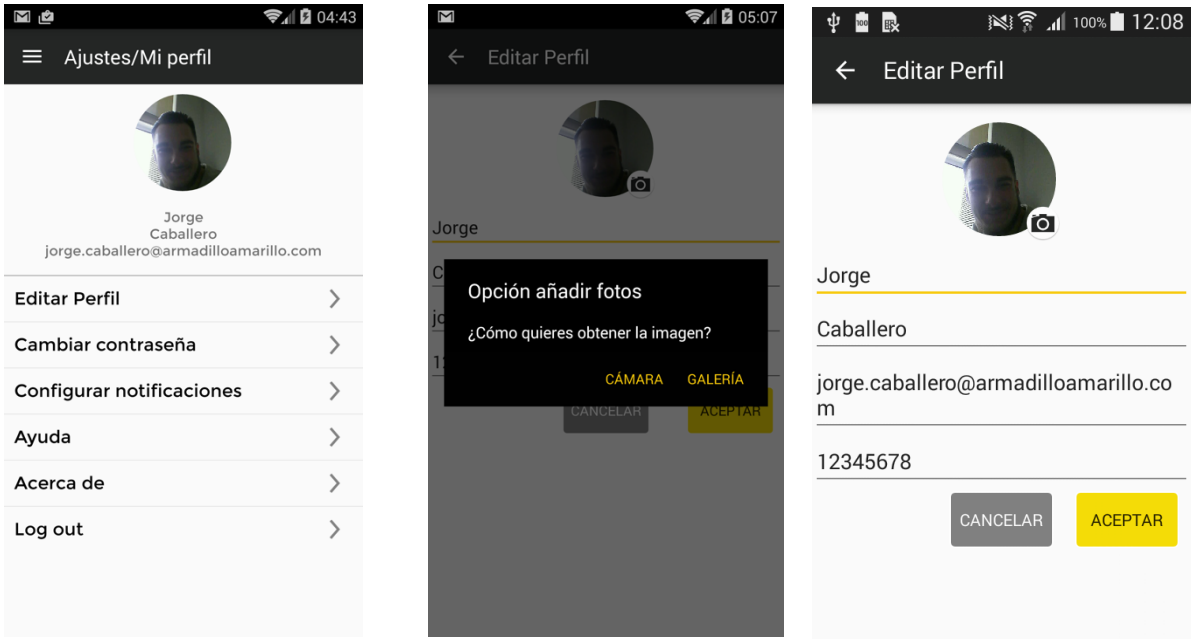

Ilustración 30 – Perfil. Vista principal, cambiar foto de perfil y editar perfil – Diseño final de la aplicación.

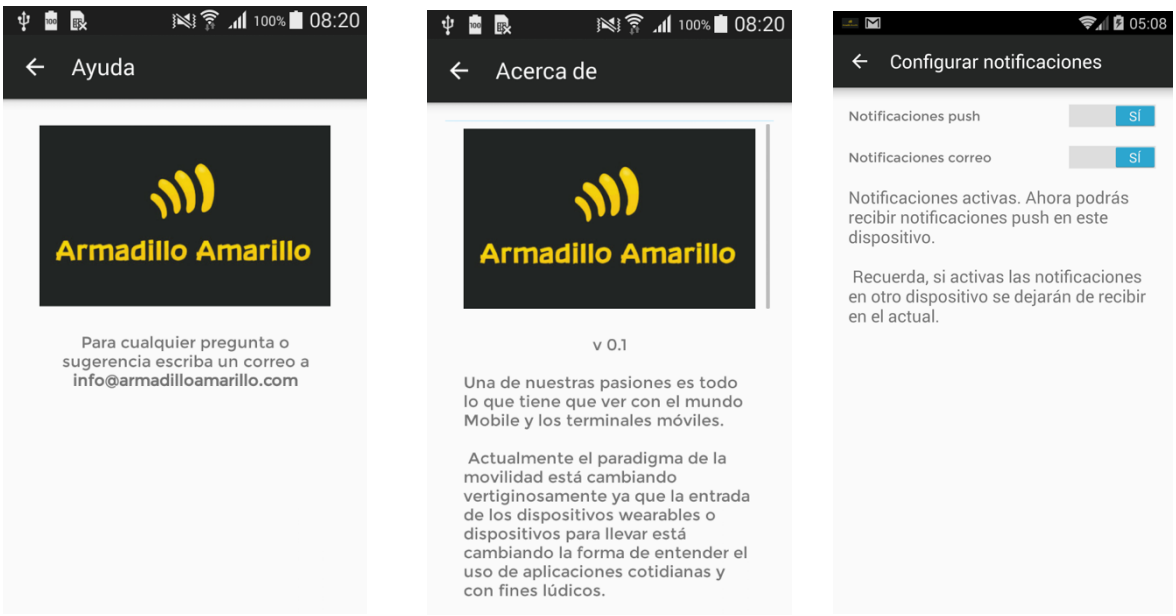

Ilustración 31 – Perfil. Ayuda, Acerca de y Configurar notificaciones – Diseño final de la aplicación.

3.3.7 Panel principal

La vista del panel principal es la **Home** de la aplicación, es decir, la primera pantalla que ve el usuario al acceder a la misma. Muestra un listado desplegable de los principales módulos con los *últimos 3 registros* insertados para el usuario. Si hubiera menos de 3 registros muestra los registros existentes y si no hubiera ningún registro mostraría un texto informativo indicando al usuario que no tiene ningún registro todavía.

Ilustración 32 – Panel Principal –Diseño inicial de la aplicación.

En esta vista también encontramos un acceso directo a crear un nuevo registro, tanto para añadir una nueva hora, mensaje como recurso.

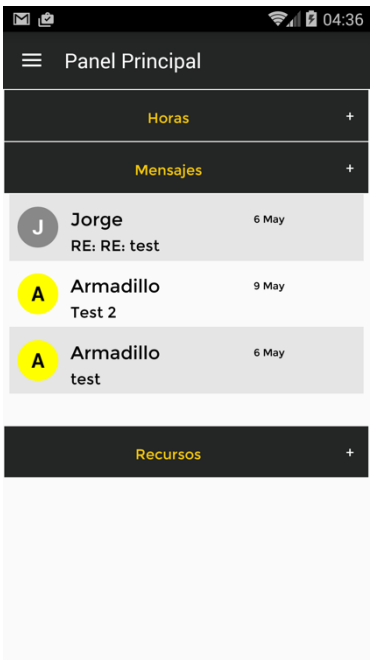

Ilustración 33 – Panel Principal – Diseño final de la aplicación.

3.3.8 Notificaciones en el dispositivo

El usuario tiene la posibilidad de activar **las notificaciones "push"** en su dispositivo a través de "Configurar notificaciones" en su perfil. De esta manera el usuario da de alta su dispositivo en **GCM** recibiendo así una notificación cada vez que reciba un nuevo mensaje.

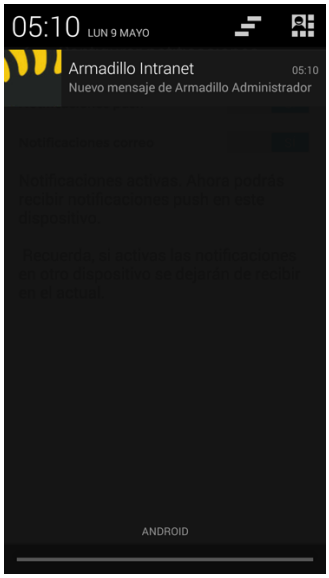

Ilustración 34 – Ejemplo de notificación push cuando se recibe un mensaje.

3.3.9 Funcionalidad del wearable

Cuando el usuario recibe una *notificación push* en su dispositivo esta se reproduce en el wearable sincronizado con anterioridad. De esta manera, el usuario podrá ver e interactuar con el reloj directamente.

Ilustración 35 – Nueva notificación y detalle del mensaje.

El usuario puede ver la notificación y con un sencillo gesto de *swipe lateral* puede acceder a varias funcionalidades desde el reloj. Véase en la imagen anterior el *detalle del mensaje*.

Así mismo también puede ir a "*Ver detalle*", que abrirá el detalle del mensaje en el a aplicación móvil y a "*Responder*", dónde se le mostrará varios *textos predefinidos* para contestar y la posibilidad de hacerlo mediante *comando de voz*.

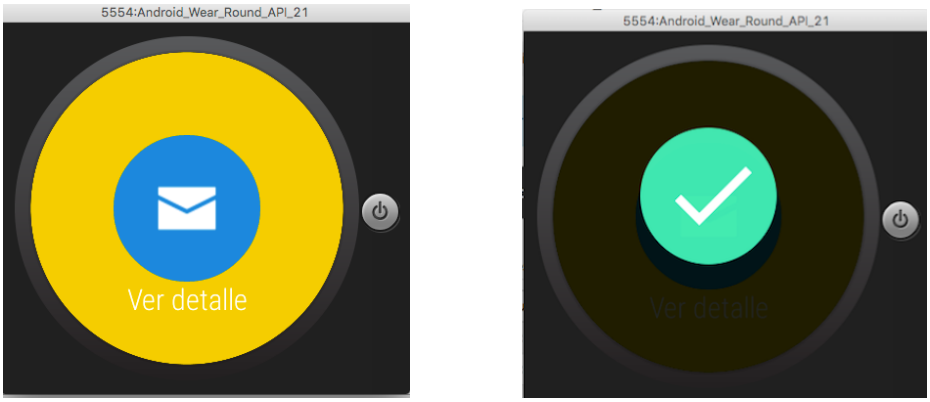

Ilustración 36 – Funcionalidad de Ver detalle.

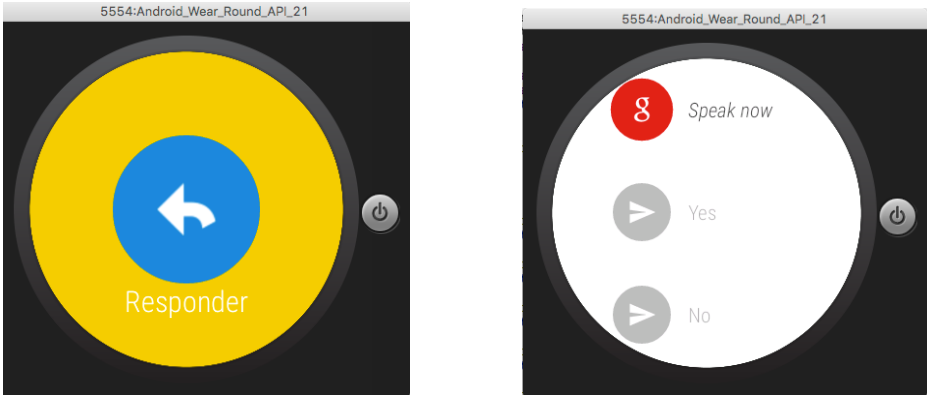

Ilustración 37 – Funcionalidad de Responder (Comando de voz y texto predefinido).

Ilustración 38 – Funcionalidad de Responder (Enviando respuesta).

Android Wear tiene por defecto la dos funcionalidades de "*Open on phone*", que llevará al usuario al panel principal de nuestra aplicación, y *"Block app*", que bloqueará todas las notificaciones push de dicha aplicación en el reloj, siguiendo mostrándolas en el dispositivo móvil.

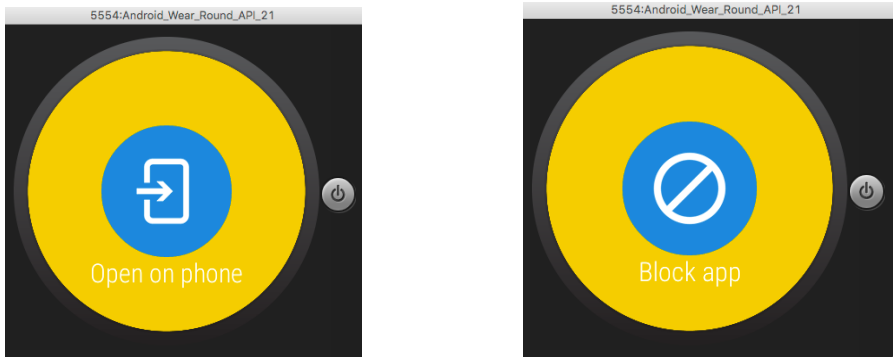

Ilustración 39 – Funcionalidad de Open on phone y Block app.

3.4 Características del usuario objetivo

El usuario final de la aplicación es toda aquella persona que pertenezca a la empresa gestionada con la Intranet web que esté en disposición de un dispositivo móvil Android.

Esta aplicación está orientada al trabajador de la empresa, ya que se han desarrollado los módulos pertenecientes al usuario de carácter de empleado, dejando de lado en un principio los módulos de gestión pertenecientes al administrador de la Intranet, cuyas funcionalidades solo se podrán realizar desde el propio servidor web.

3.5 Estimación del proyecto y presupuesto

La estimación de la duración del proyecto se realizó durante la primera fase de conceptualización. Esta planificación se dividió en estimación de **diseño de la aplicación**, estimación del desarrollo de la **funcionalidad de la aplicación y wearable** y estimación de desarrollo de la **funcionalidad de los servicios** de la API (backend). Para ilustrar la planificación del proyecto utilizamos el diagrama de Gantt, en el cual se dividen las fases del proyecto en Sprints como se indica en la introducción de este documento.

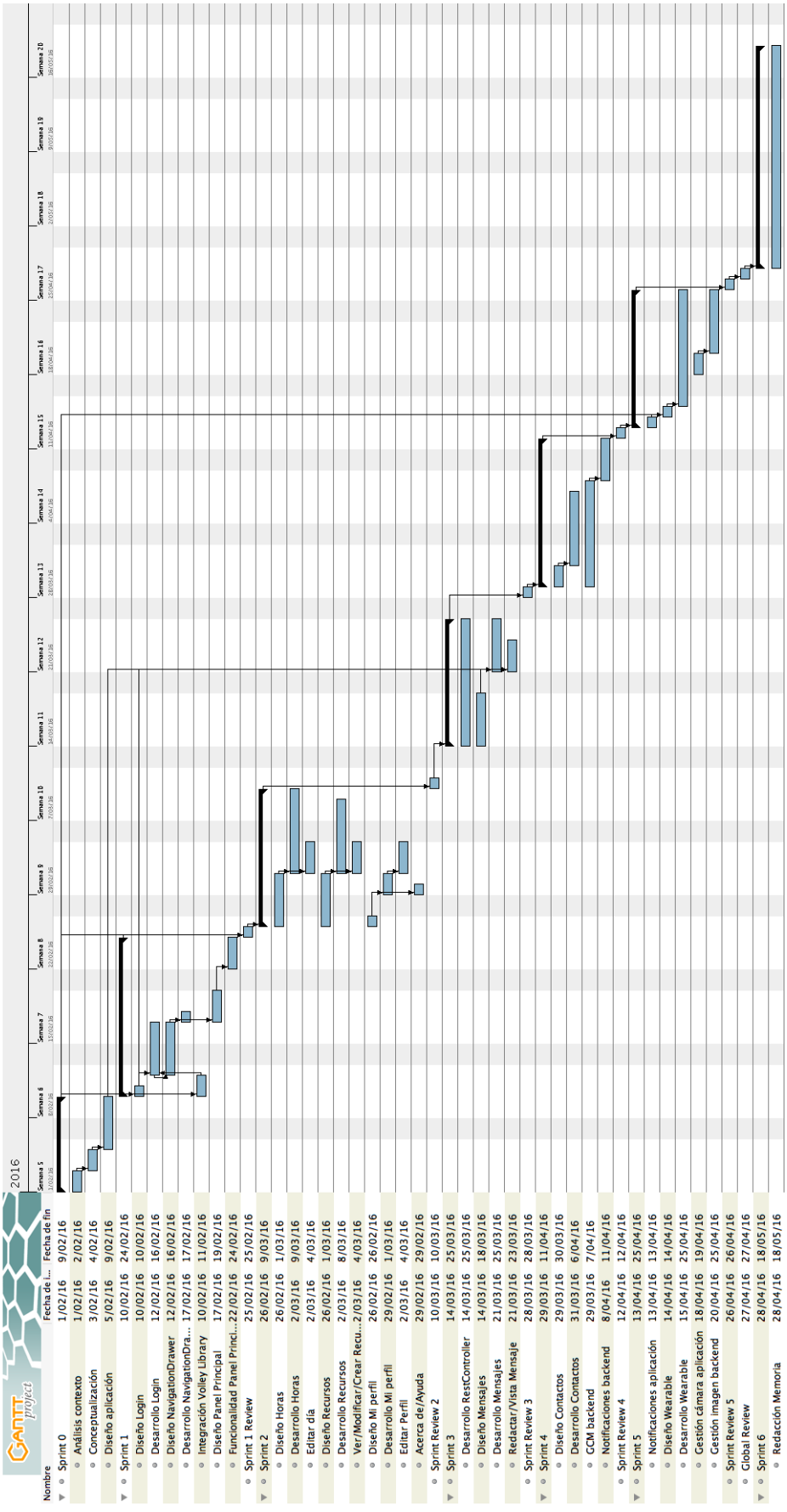

Ilustración 40 – Diagrama de Gantt.[12]

[12] Diagrama de Gantt generado con la herramienta GanttProject

A continuación se expone el presupuesto detallado del proyecto contando con los costes directos e indirectos:

Personal	Grado	Categoría	Horas	Coste por hora (euros)	Coste (Euros)
Jorge Caballero Amo	Ingeniero Junior	Diseño	10	35	350
		Desarrollo	305	35	10675
		Redacción documentación	75	35	2625
Javier Fernández	Ingeniero Senior	Desarrollo y QA	40	50	2000
				TOTAL (Euros)	**15650**

Tabla 62 – Presupuesto del personal.

Descripción	Coste (Euros)	Tiempo de uso (meses)	Factor de utilización (0.1-1)	Período de vida útil (meses)	Coste Imputable (Euros)
MacBook Pro	1279	4	0.9	96	47,9625
Samsung Galaxy Core 2	120	4	0.9	24	18
LG Watch Urbane	349	1	0.4	36	3,8
				TOTAL (Euros)	**69,7625**

Tabla 63 – Costes amortización de equipos.

El coste asociado a la amortización de los equipos ha sido calculado por medio de la siguiente fórmula:

$$\frac{Coste * Tiempo\ de\ uso * Factor\ de\ utilizacion}{Periodo\ de\ vida\ util}$$

Por tanto, con los datos extraídos podemos concluir que el presupuesto del proyecto es el siguiente:

Tipo	Coste (Euros)
Personal	15650
Amortización	69,7625
TOTAL	**15719,7625**

Tabla 64 – Presupuesto total.

3.6 Metodología de trabajo

La metodología utilizada para llevar a cabo este proyecto ha sido la metodología **SCRUM**. Esta metodología se basa en un desarrollo incremental solapando las fases de desarrollo en lugar de realizar una tras otra en un ciclo secuencial.

Este método persigue el mejor resultado de un proyecto y la *agilidad* en la sucesión de tareas, siempre teniendo en cuenta un conjunto de buenas prácticas. Este proceso subdivide el proyecto en una serie de "deliverys" o "sprints" que se deben entregar al cliente para que éste vaya viendo el avance del proyecto de manera continuada.

El desarrollo del producto se divide en **Épicas**, estas son los requisitos generales del proyecto, que a su vez se dividen en subtareas denominadas **Historias de usuario**, que se definen por unos requisitos específicos. En cada **Sprint** se incluirán una serie de Historias de usuario que deben ser realizadas antes de que termine el mismo.

En nuestro caso, al no tener un cliente como tal (al tratarse de un producto de la empresa), estas entregas se realizan cada dos semanas al **Product Owner**, que para Armadillo Intranet es Carlos Cuadrado (cofundador de Armadillo Amarillo). El **Scrum Master** (Carlos Cuadrado) es el encargado de supervisar que el equipo avanza de manera regular en el desarrollo de las tareas y de facilitar el desarrollo de alguna tarea si ésta está bloqueada.

Durante el avance del Sprint, cada desarrollador elige la historia de usuario que quiere realizar, es decir, son auto asignadas, no son asignadas por el Scrum Master del proyecto.

Al final de cada Sprint se realizará una “review” de las tareas que se han desarrollado durante el mismo. Esta revisión consiste en hacer una serie de pruebas sobre las historias de usuario que se han llevado a cabo (testing y QA) y en enseñar el prototipo de la aplicación al responsable.

Para que el Scrum Master tenga una visión global del proyecto utilizamos el gestor de proyectos **JIRA**[13]. Este gestor tiene una pizarra que se compone de varias columnas con las tareas correspondientes al Sprint:

- *Por hacer*. Para las tareas que deben realizarse.
- *En curso*. Para las tareas que ya están asignadas y están realizándose.
- *Bloqueado*. Para las tareas que están bloqueadas.
- *QA*. Para las tareas que tienen que pasar un testing y comprobar que cumplen con su funcionalidad.
- *Listo*. Para las tareas que están finalizadas.

[13] JIRA Software https://es.atlassian.com/software/jira

Siguiendo este procedimiento, ningún miembro del equipo realizará una tarea que estaba siendo realizada por otra persona, se tiene una visión del avance del proyecto y se ponen unas metas asumibles a corto plazo para que el proyecto no se quede parado.

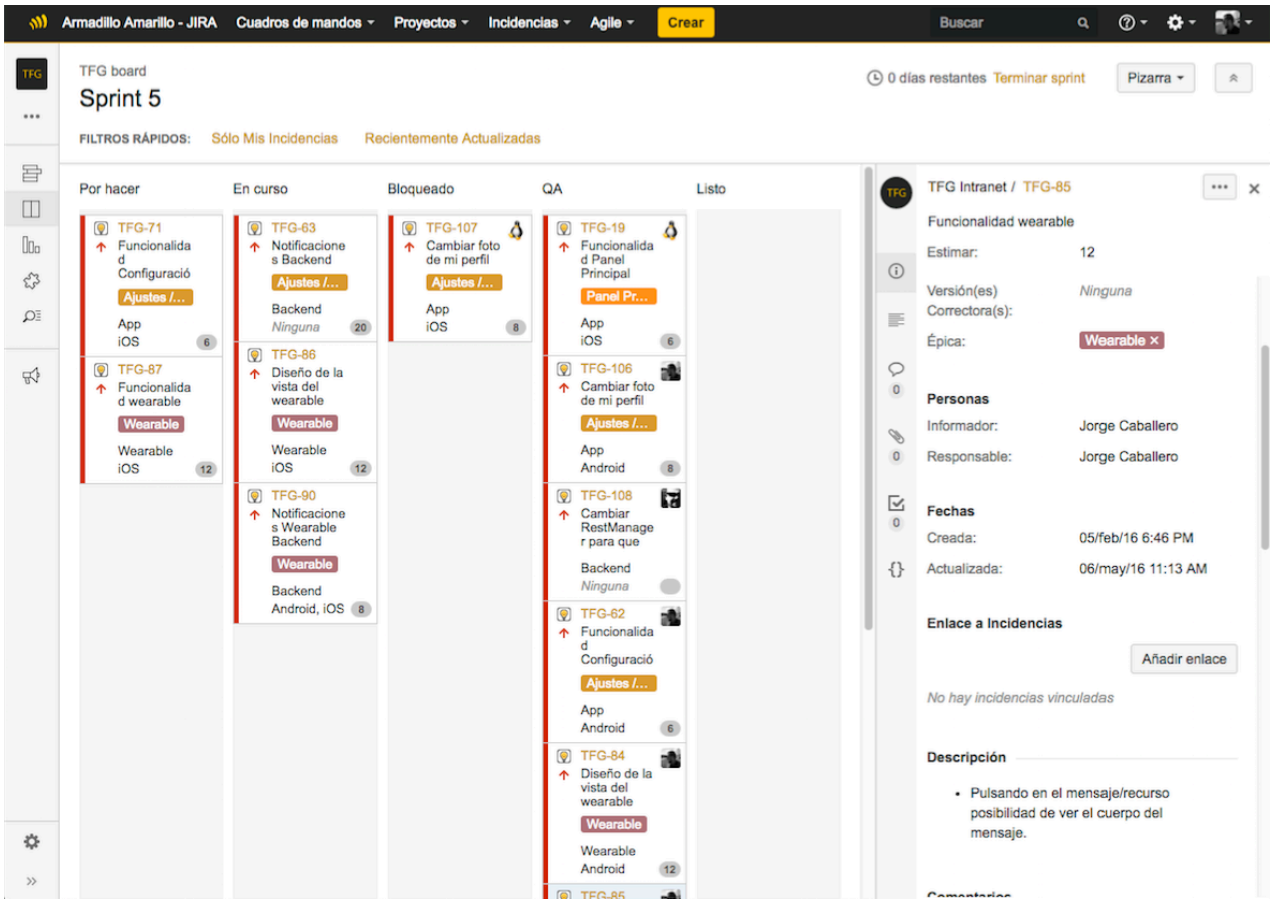

Ilustración 41 – JIRA Sprint 5

Por último, varios miembros del equipo pueden trabajar en local sobre el mismo documento, lo que implica problemas a la hora de subir el código fuente al servidor. Para solventar este problema se utiliza el software de control de versiones **Git**[14], en concreto, se utiliza el cliente **SourceTree**[15].

Este software de control de versiones permite trabajar en diferentes ramas individuales para cada usuario, las cuales nacen de la rama de desarrollo “develop”, que a su vez nace de la rama "master".

El procedimiento seguido para trabajar con Git es utilizando **Git Flow**, que genera una rama individual que nace de develop en la cual el desarrollador puede realizar todos sus cambios (generalmente utilizada para cada tarea de usuario). Cuando acabe el desarrollo de la misma, se procede a realizar el *merge* de su código en la rama de develop. El *merge* consiste en juntar el código

[14] Sistema de control de versiones Git: https://git-scm.com/
[15] Cliente de Git SourceTree: https://www.sourcetreeapp.com/

existente en una rama de desarrollo con otra rama. De esta manera no destruiremos el trabajo de otro compañero y evitaremos conflictos de código.

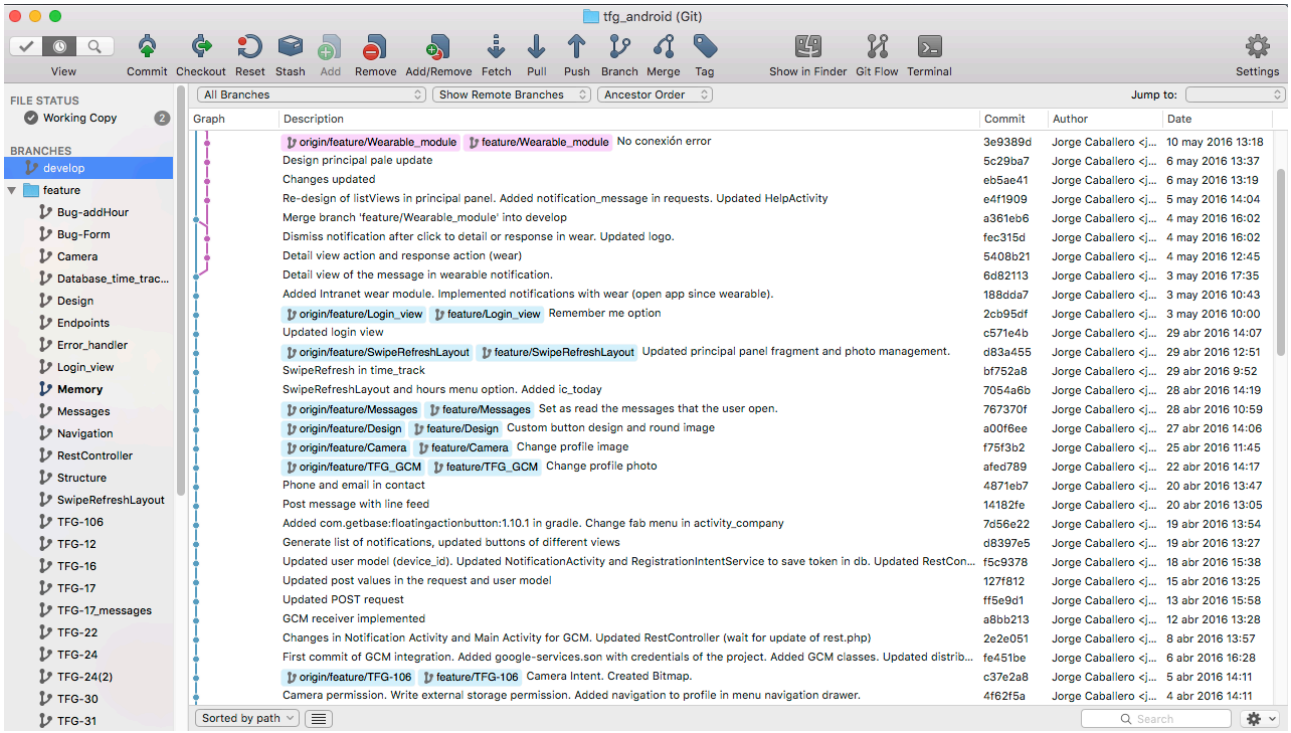

Ilustración 42 – SourceTree – Cliente de escritorio de Git.

Capítulo 4. Desarrollo de la aplicación

4.1 Diagramas de clases

La estructura de clases del proyecto han sido divididas en varios paquetes con el objetivo de organizar las funcionalidades y el entendimiento para otros desarrolladores ya que se trata de un proyecto de grandes dimensiones y en el que en un futuro pueden participar otros compañeros.

Actividades: engloba todas las actividades del proyecto. Todas las actividades siguen la nomenclatura *NameActivity*, dónde *Name* es un nombre específico que describa de manera lógica la funcionalidad de la actividad.

- AboutActivity
- AddHourActivity
- AddResourceActivity
- ChangePasswordActivity
- CompanyActivity
- DetailContactActivity
- DetailMessageActivity
- EditHourActivity
- EditProfileActivity
- EditResourceActivity
- HelpActivity
- MainActivity
- NavigationDrawerActivity
- NewMessageActivity
- NotificationActivity
- ReplyActivity
- ResourceGroupActivity

Adaptadores: adaptadores necesarios para construir las listas con una estructura diferente dependiendo del objeto que se esté mostrando. Todas los adaptadores siguen la nomenclatura *NameAdapter*, dónde *Name* es un nombre específico que describa de manera lógica la funcionalidad del adaptador.

- CompanyAdapter
- ContactAdapter
- EmailAdapter
- ListViewAdapter
- MessageListViewAdapter
- ResourceListViewAdapter
- TimeTrackListViewAdapter

Custom: clases que personalizan diferentes aspectos de la aplicación, en este caso la imagen de perfil y el formato de texto (para poder utilizar el tipo de texto corporativo de las diferentes empresas).

- CustomImageView
- CustomTextView

Fragments: engloba todos los fragments de la aplicación (módulos que se muestran desde el panel principal y fragment utilizado para elegir una hora en el módulo de imputación de horas). Todas los fragments siguen la nomenclatura *NameFragment*, dónde *Name* es un nombre específico que describa de manera lógica la funcionalidad del fragment.

- DatePickerFragment
- HoursFragment
- HowFragment
- MessagesFragment
- MyProfileFragment
- PanelFragment
- ResourcesFragment

GCM: gestión de la conexión con *Google Cloud Messaging*.

- MyGcmListenerService
- MyInstanceIDListenerService
- QuickstartPreferences
- RegistrationIntentService

Listeners: escuchadores de la aplicación. En nuestro caso solo tenemos uno para recargar al hacer “scroll down” (OnScrollUpDownListener) en varias vistas pero en un futuro podrían ser necesarios más.

Modelos: engloba los diferentes modelos de datos que se utilizan en la aplicación, todos tienen la misma estructura que las tablas de la base de datos que representan.

- Company
- Contact
- Message
- Resource
- ResourceGroup
- ResourceType
- Time_track
- Time_trackType
- User

Servicios: incluye la configuración de los *endpoints* del servidor (servicio Config) a los que apunta la aplicación para poder migrar el servidor sin que sea complicado para la aplicación.

Sincronización: clases que gestionan los objetos y la cola de peticiones cuando se realiza la integración con Volley.

- AppController
- LruBitmapCache

Utils: carpeta que contiene las clases e interfaces encargadas de la comunicación con el servidor. Tenemos la interfaz *AsyncResponse* con los métodos:

- *processFinish(JSON output, int type)* dónde output es la respuesta del servidor y type es un identificador que sirve para identificar distintas peticiones dentro de una misma clase.
- *imageFinish(Bitmap output)* dónde output es el mapa de bits que devuelve el servidor correspondiente a la foto de perfil del usuario.

Estos métodos tendrán que ser definidos en todas las clases que implementen la interfaz *AsyncResponse* aunque solo se utilice uno de los métodos citados.

- AccountUtils
- AsyncResponse
- ImageProfile
- ListUtils
- RestController

Todas las vistas de la aplicación necesitan conexión a internet para poder realizar la petición al servidor de la empresa con el objetivo de obtener los datos necesarios para mostrarlos.

Para ello se ha implementado una clase **RestController.java** (dentro de la carpeta *Utils* indicada anteriormente) que extiende de *AsyncTask* para realizar la tarea de forma asíncrona. De esta manera realizamos la petición en otro hilo de ejecución que no sea el principal con el objetivo de que la experiencia del usuario sea lo más cómoda posible.

Esta clase es llamada cada vez que se quiere hacer una petición al servidor. Está estructurada de tal manera que organiza las peticiones con el objetivo de hacerlo de la manera más escalable posible, es decir, que se puedan ampliar los servicios de la API sin la necesidad de generar muchas líneas de código en la aplicación Android.

4.1.1 Estructura de las clases .java

Cada clase del proyecto ha sido implementada para una funcionalidad específica pero todas ellas tienen un patrón de construcción para que sea más sencillo para los desarrolladores de la aplicación saber dónde buscar algo específico dentro de cada clase aun no siendo ellos quien lo hallan implementado. A continuación explicaremos la estructura que siguen las clases de manera global:

Activities:

- Controlan la lógica de la aplicación.
- *onCreate(Bundle):* método al que se llama cuando se inicia la actividad. Se le indica que vista tiene que inflar. Generalmente llamamos a los siguientes métodos dentro de éste con el objetivo de hacer el código más ordenado y organizado:
    - *buildUser()*: crea el modelo de usuario para esa actividad.
    - *initViews()*: referencia las vistas del xml de la actividad para poder interactuar programáticamente con ellos.
    - *setListeners()*: referencia los escuchadores a los elementos inicializados en el método initViews.
    - Métodos para realizar peticiones específicas de la actividad: *getEmailInfo(), requestResourceType(), requestResourceGroup(), getContactValues(),* etc.

Adaptadores:

- Todos ellos extienden de *BaseAdapter*. Sirven para construir listas de manera dinámica en la aplicación.
- Se componen de un constructor cuyos parámetros de entrada son el contexto desde dónde se crea el adaptador y la lista de información que tiene que ser pintada.
- El adaptador se encarga de dar formato y construir las filas de la lista de manera personalizada.

Custom:

- Clases creadas específicamente para personalizar el diseño la aplicación. Estas clases sustituyen a los elementos del xml. Por ejemplo, en vez de construir un elemento de la clase *ImageView* (por defecto en Android) construiremos un elemento de la clase *CustomImageView* creada por nosotros (en este caso para dar un formato redondo a la imagen con un tamaño adaptado a la vista).

Fragments:

- *onCerateView(LayoutInflater, ViewGroup, Bundle):* hace las funciones del onCreate de una actividad. También se le indica qué layout debe inflar para la vista. Se realiza la llamada a los mismos métodos que en una actividad excepto a buildUser(), ya que todos los Fragments son cargados dentro de *NavigationDrawerActivity* (actividad contenedora), de esta manera se puede acceder a la información del usuario a través de esta actividad.

Modelos:

- Constructor: específico para cada modelo de datos, que tendrá diferentes variables dependiendo de la estructura de la tabla de la base de datos a la que haga referencia.
- *Getters y setters*: métodos para acceder o modificar la información específica de los modelos desde los controladores de la aplicación.

Dadas las dimensiones del proyecto no se puede hacer un desglose específico de todos los métodos de las clases de las que se compone. Realizamos un estudio del flujo de datos que sigue la aplicación en cada vista.

4.1.2 Vista Login

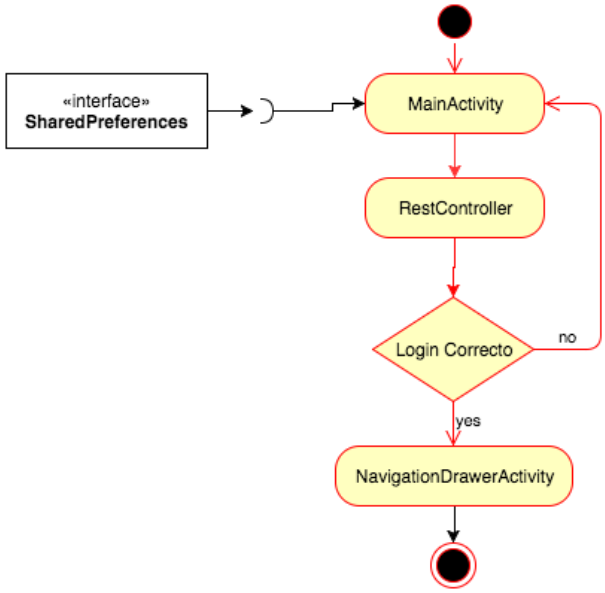

Ilustración 43 – Diagrama de flujo - Login[16]

- **MainActivity**: controla la lógica de la vista. Se hace uso de *SharedPreferences* para almacenar las credenciales si quieren ser recordadas y para almacenar los datos del usuario.
- Si el login resulta correcto se abre la siguiente vista, *NavigationDrawerActivity* con la vista del panel principal.
- Si el login resulta incorrecto (credenciales inválidas o no existe conexión a internet): se muestra el mensaje de error al usuario.

4.1.3 Menú lateral

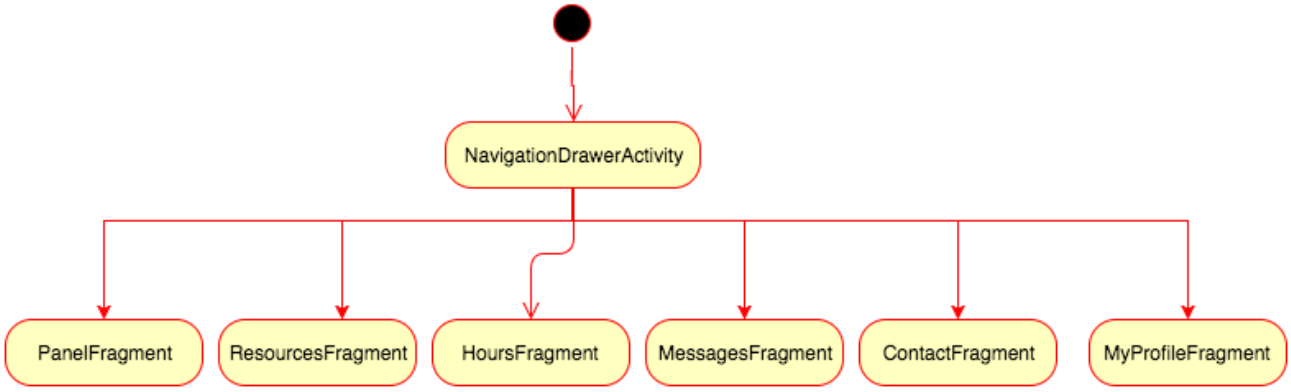

Ilustración 44 – Diagrama de clases – Menú Lateral

[16] Diagramas de clases generados con la herramienta online draw.io: https://www.draw.io/

- **NavigationDrawerActivity**: actividad principal de la aplicación (no confundir con el *MainActivity*, actividad con la se inicia la aplicación). Se compone del menú lateral y el contenedor de Fragment para ir inflando las diferentes vistas principales de la aplicación según los intereses del usuario.
- Por defecto se infla la vista del panel principal. La parte superior del menú lateral carga la información de la sesión del usuario junto con su foto.

4.1.4 Panel Principal

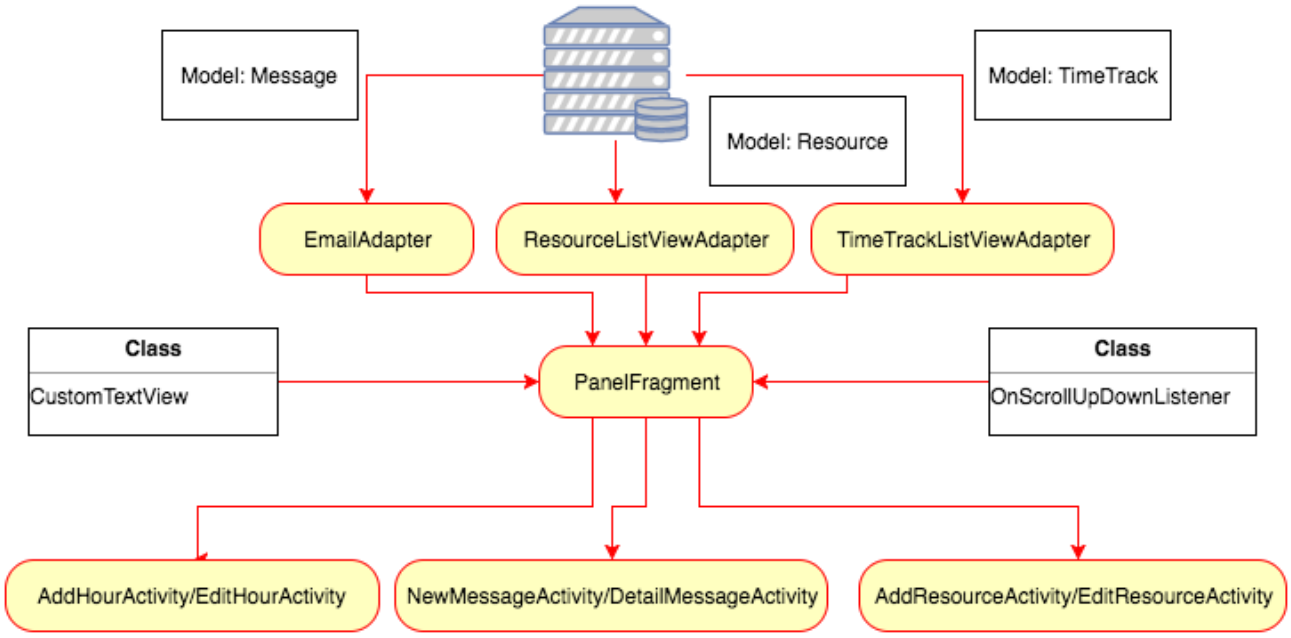

Ilustración 45 – Diagrama de clases – Panel Principal

- **PanelFragment**: controla la lógica de la vista del panel principal. Implementa *AsyncResponse* para realizar las peticiones al servidor. Carga las listas de objetos ayudado de la clase *ListUtils* que adapta el tamaño de la lista a la pantalla del dispositivo y los tres adaptadores de listas: *EmailAdapter*, *ResourceListViewAdapter* y *TimeTrackListViewAdapter*.
- Se implementa un escuchador en los botones que dependiendo de la interacción del usuario cargará las clases: *AddHourActivity* (nueva hora), *NewMessageActivity* (nuevo mensaje) o *AddResourceActivity* (nuevo recurso).
- Se implementa un escuchador en las listas de objetos que dependiendo del objeto con el que interaccione el usuario cargará: *EditHourActivity* (detalle de la hora seleccionada con posibilidad de editarla), *EditResourceActivity* (detalle del recurso con posibilidad de editarlo) y *DetailMessageActivity* (detalle del mensaje).
- Se utilizan las clases *CustomTextView* para los títulos de las secciones.
- Se utiliza *OnScrollUpDownListener* para recargar la información de la vista.
- Se utilizan los modelos de datos *Message*, *Resource* y *TimeTrack*.

4.1.5 Recursos

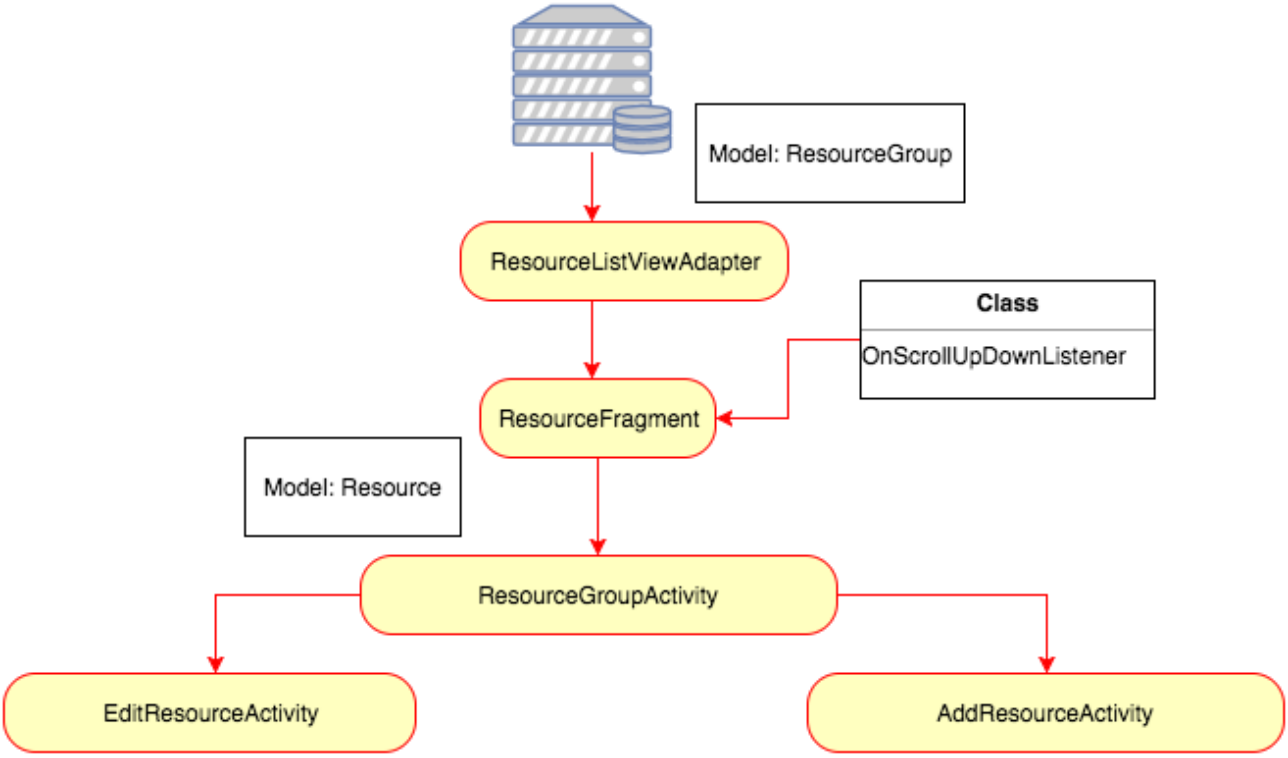

Ilustración 46 – Diagrama de clases – Recursos

- **ResourcesFragment**: controla la lógica de la vista inicial de los recursos. Implementa *AsyncResponse*. Realiza la petición al servidor de *requestGroupResources()* para poder cargar la lista con los grupos de recursos accesibles para el usuario.
- Se implementa un escuchador en la lista, que lanza *ResourceGroupActivity* con el detalle de los recursos de grupo dependiendo del grupo que ha pulsado el usuario.
- Se utiliza *OnScrollUpDownListener* para recargar la información de la vista.
- **ResourceGroupActivity**: realiza la petición de *requestResourcesForGroup()* para cargar en la lista los diferentes recursos de grupo. Implementa un escuchador en la lista que lanza *EditResourceActivity*. Implementa un escuchador de un botón que lanza *AddResourceActivity* para generar un nuevo recurso de grupo.
- **AddResourceActivity** y **EditResourceActivity**: ambas actividades realizan una petición para recuperar los diferentes grupos de recurso y los tipos de recurso, y los inflan en un *Spinner*. Ambas controlan una vista de un formulario, la actividad de añadir recurso lo muestra vacío con los datos por defecto y la actividad de editar recurso muestra el detalle del recurso seleccionado con la posibilidad de modificarlo por el usuario o borrar el recurso.
- En todas las actividades se utiliza el modelo de datos *Resource*, *ResourceGroup* y *ResourceType*.

4.1.6 Registro de horas

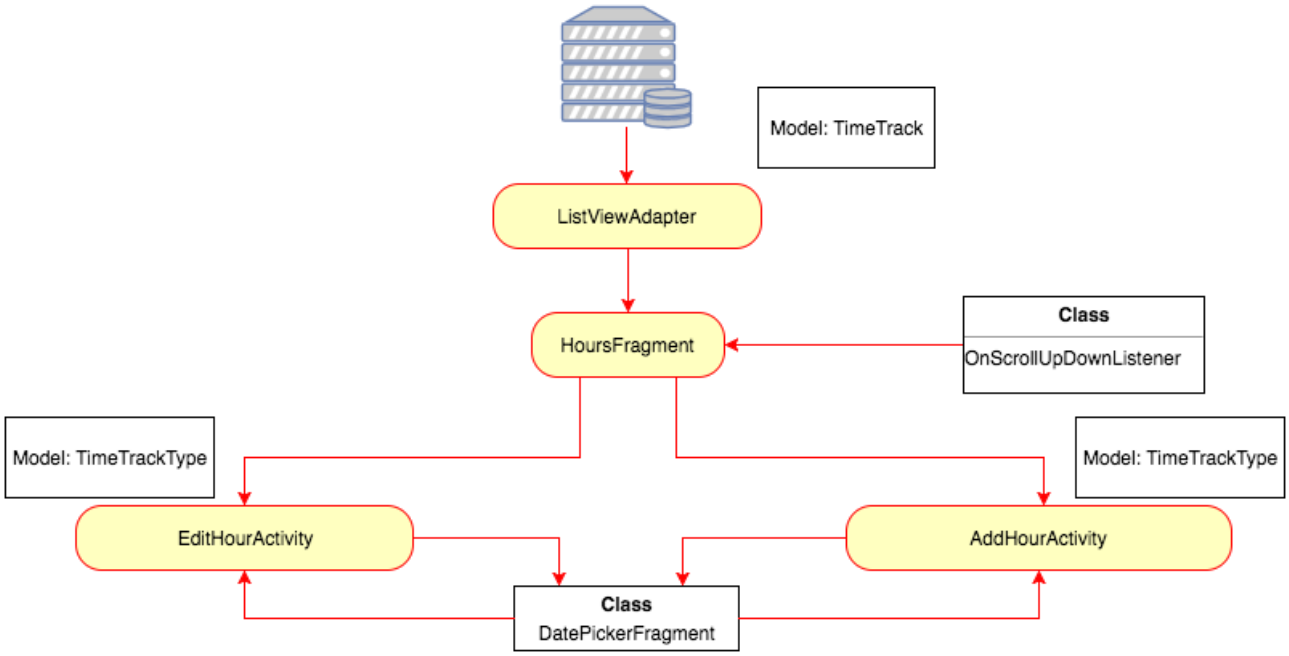

Ilustración 47 – Diagrama de clases – Registro de horas

- **HoursFragment**: controla la lógica de la vista principal del registro de horas, que se compone de un calendario diseñado con *Material Design* y una lista de horas en la parte inferior. Implementa *AsyncResponse* para realizar las peticiones al servidor. Por defecto se inicializa el día seleccionado al día de hoy. Se realiza la petición *requestHours* con el día seleccionado para recuperar las horas imputadas. Estas horas son cargadas en la lista mediante *ListViewAdapter*.
- Se implementa un escuchador en el calendario, que realiza la petición de *requestHours* dependiendo del día seleccionado en el calendario por el usuario y actualizando la lista de horas.
- Se implementa un escuchador en la lista para lanzar *EditHourActivity* dependiendo de la hora seleccionada por el usuario.
- Se implementa un escuchador para el botón de añadir hora que hace que aparezca o desaparezca dependiendo del scroll de la lista para mejorar la experiencia de usuario. Si el usuario lo pulsa se lanzará *AddHourActivity*.
- **AddHourActivity** y **EditHourActivity**: realizan las peticiones *requestTimeTrackType* y *requestTimeTrackProject* para recuperar los proyectos y el tipo de imputación. Esta información será inflada en el *Spinner*. Se lanzará *DatePickerFragment* para que el usuario pueda modificar la fecha de imputación. Ambas controlan una vista con un formulario, en la actividad de añadir nueva hora se muestra vacío y en la actividad de editar hora se muestra con el detalle de la hora con la posibilidad de modificarla o borrarla.
- Se utiliza el modelo de datos *Time_track* y *Time_trackType*.

4.1.7 Mensajes

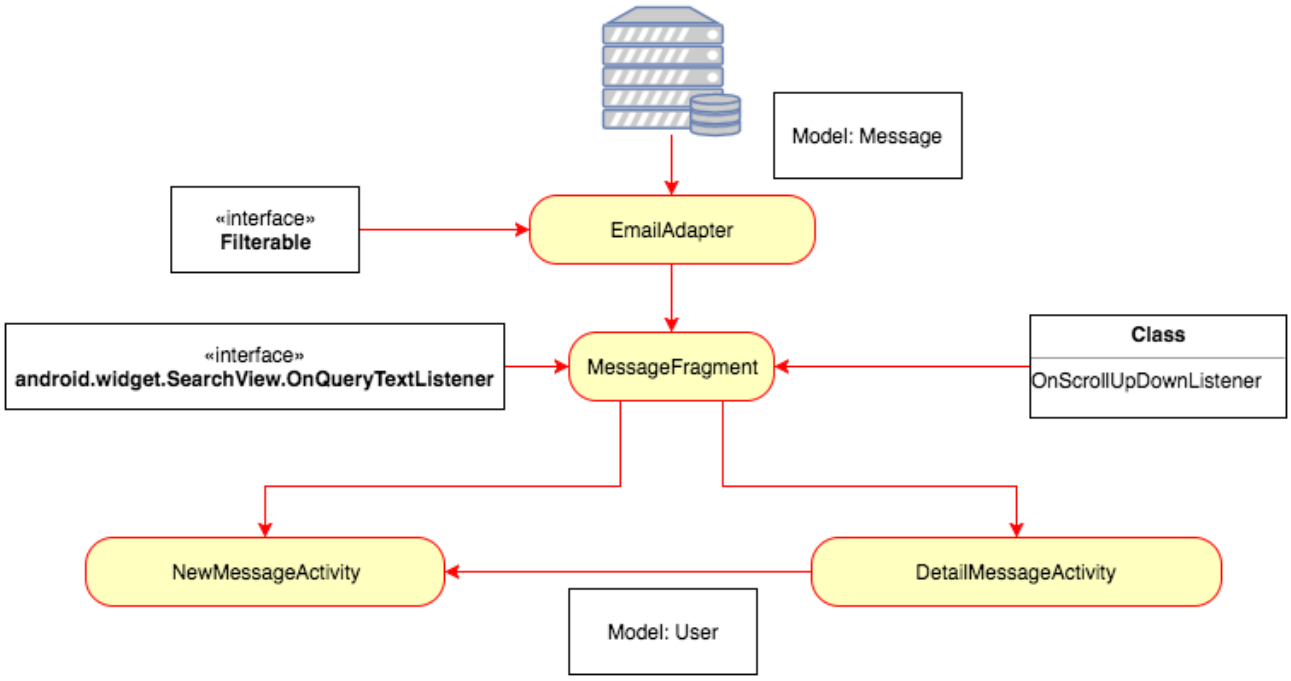

Ilustración 48 – Diagrama de clases – Mensajes

- **MessageFragment**: controla la lógica de la vista principal de mensajes. Implementa *AsyncResponse* para realizar las peticiones y *android.widget.SearchView.OnQueryTextListener* para la implementación del buscador en la parte del menú del Fragment. Realiza la llamada al método *requestMyMessages* para recuperar todos los mensajes recibidos y *requestSentMessages* para recuperar los mensajes enviados. Carga los mensajes en la lista mediante *EmailAdapter*. Por defecto muestra la bandeja de entrada del usuario.
- **EmailAdapter** implementa la clase *Filterable*, dando la posibilidad de filtrar los objetos que contiene en la lista. Es de esta manera como actualizamos de forma automática la lista cuando realizamos la búsqueda implementada en el menú superior.
- Implementa un escuchador sobe la lista, que lanza *DetailMessageActivity* con los detalles del mensaje que el usuario haya pulsado.
- Implementa un escuchador en el botón de "Nuevo mensaje" que lanza *NewMessageActivity*. Este botón también tiene otro escuchador que hace que aparezca o desaparezca dependiendo del scroll que tiene la lista para mejorar la experiencia de usuario.
- Se utiliza *OnScrollUpDownListener* para recargar la información de la vista.
- **DetailMessageActivity**: controla la lógica de la vista de detalle del mensaje. Carga la información del mensaje obteniéndola como extra del Intent del cual ha sido llamado mediante *getIntent()* y la infla en la vista. Si el mensaje no ha sido leído con anterioridad y el usuario es el destinatario se realiza la petición de marcar como leído el mensaje. El detalle del mensaje se carga en elementos *CardView* de Android. Se realizará la petición al servidor de borrar mensaje si el usuario pulsa el botón de borrar mensaje (apareciendo un diálogo de

confirmación al usuario). Se lanza *NewMessageActivity* si se pulsa el botón de responder, que se cargará con los datos de respuesta ya completados.

- **NewMessageActivity**: controla la lógica de la vista de nuevo mensaje. Implementa *AsyncResponse*. Realiza la petición de *requestUsers* que devuelve todos los usuarios de la empresa. De esta manera el usuario cuando escribe el destinatario del mensaje automáticamente se le sugieren nombres de empleados para mejorar la experiencia de usuario. Se le mostrará al usuario diferentes diálogos de confirmación si los datos de del mensaje no son los correctos o si le falta completar algún dato.
- Se utiliza los modelos de datos *User* (para los empleados de la empresa) y *Message*.

4.1.8 Contactos

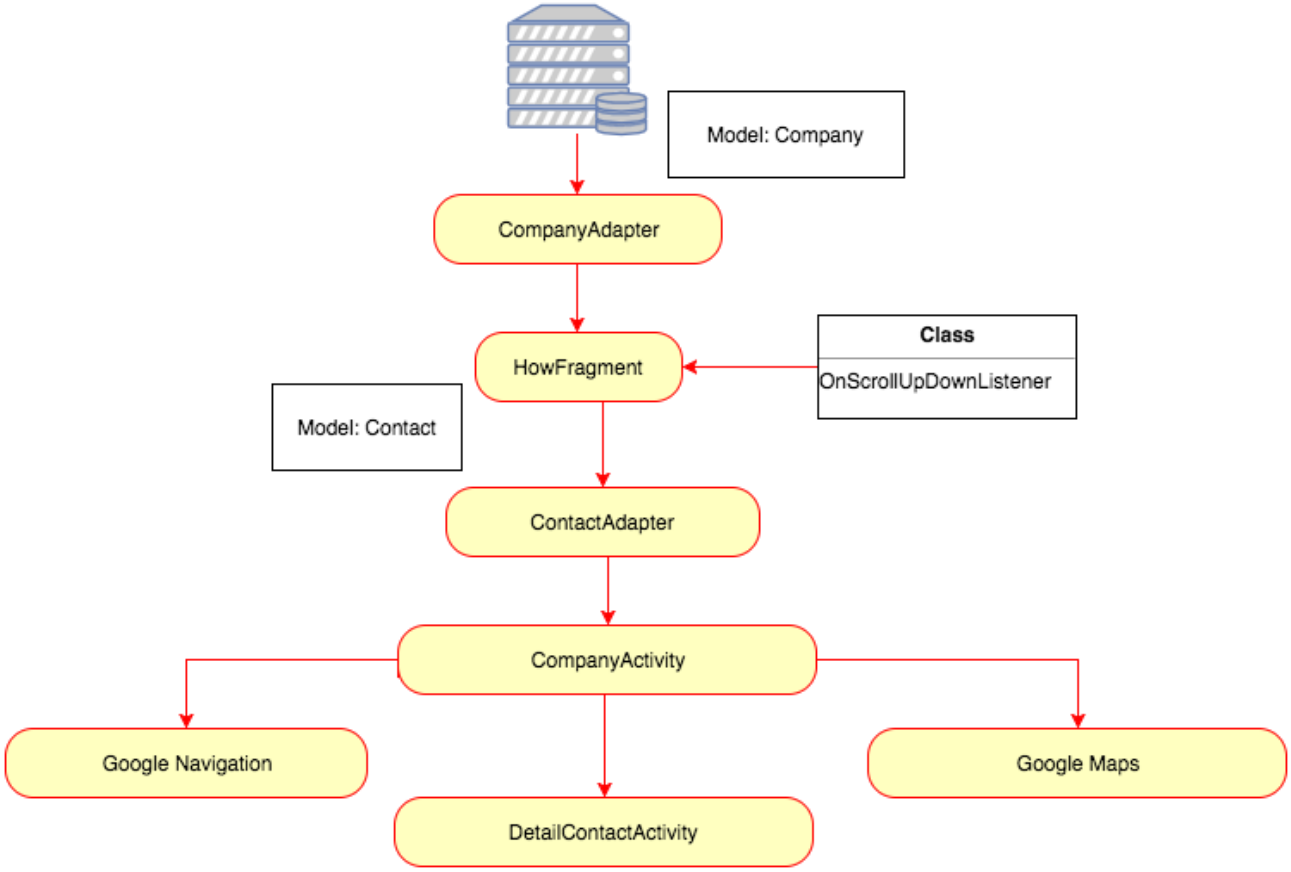

Ilustración 49 – Diagrama de clases – Contactos

- **HowFragment**: controla la lógica de la vista principal de contactos. Implementa la clase *AsyncResponse* para realizar peticiones. Realiza la petición de *requestCompanies* que devuelve las compañías para inflarlas posteriormente en una lista mediante *CompanyAdapter*.
- Implementa un escuchador sobre la lista, que lanza *CompanyActivity* con la información de la compañía dependiendo del objeto que haya pulsado el usuario.
- Se utiliza *OnScrollUpDownListener* para recargar la información de la vista.

- **CompanyActivity**: implementa *AsyncResponse*. Realiza la petición de *requestContactsForCompany* que devuelve todos los contactos para una cierta empresa. Carga la lista mediante *ContactAdapter*. Implementa varios escuchadores:
    - *Botón navegar*: muestra un diálogo al usuario y lanza una actividad externa a la aplicación que se inicia a través de *Google Navigation* pasándole como parámetros la longitud y latitud de la empresa y el medio de transporte elegido para iniciar la navegación.
    - *Botón Mapa*: lanza una actividad externa a la aplicación que se inicia a través de *Google Maps* pasándole como parámetros la longitud, latitud y nombre de la empresa.
    - Escuchador sobre la lista de contactos que lanza *DetailContactActivity* dependiendo del objeto pulsado por el usuario.
- **DetailContactActivity**: recupera la información del contacto a través del Intent desde el que ha sido llamado mediante *getIntent()* e infla la información en los elementos *CardView* del xml. Implementa un escuchador que inicia un Intent de marcado telefónico (*Intent.ACTION_DIAL*) al pulsar sobre la información del teléfono del contacto (al menos que esté sin completar). Implementa un escuchador que lanza un Intent para enviar un correo electrónico que inicia el cliente de correo que se tenga instalado en el dispositivo con la información de destinatario completada (*Intent. ACTION_VIEW*).
- Se utiliza los modelos de datos *User*, *Company* y *Contact*.

4.1.9 Mi perfil

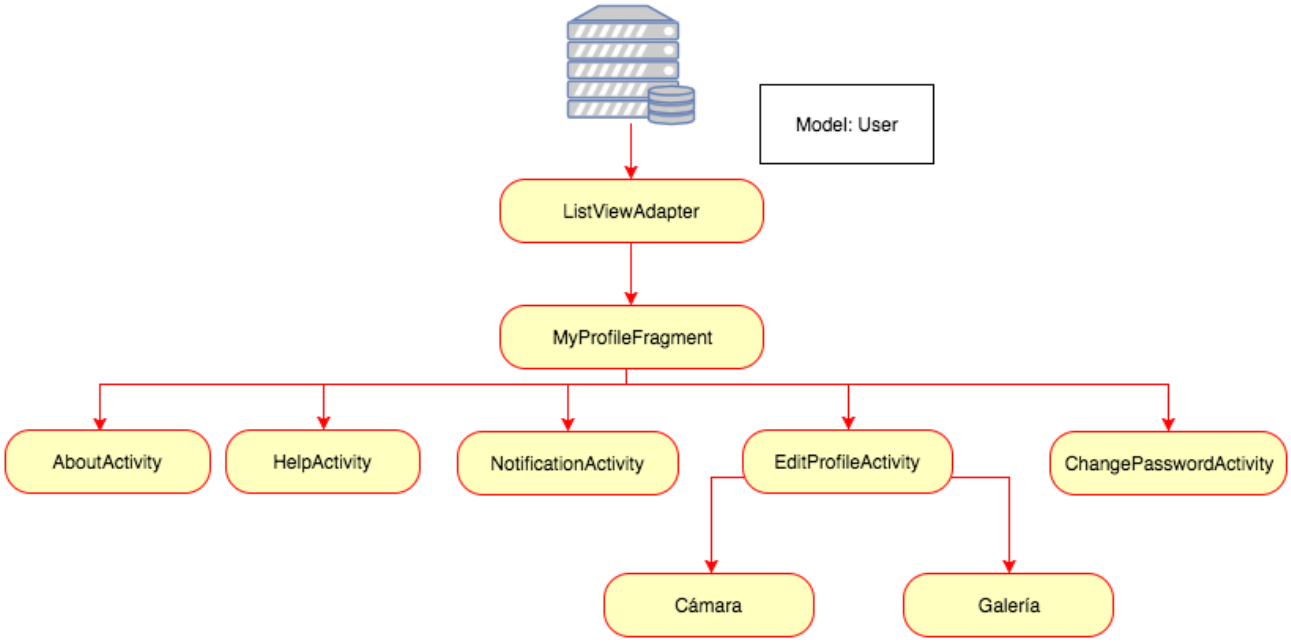

Ilustración 50 – Diagrama de clases – Mi perfil

- **MyProfileFragment**: controla la lógica de la vista principal de Mi perfil. Implementa *AsyncResponse* para la realización de peticiones. Carga la imagen de perfil mediante la

respuesta del método *imageFinish* de la interfaz *AsyncResponse*. Carga la lista de opciones mediante *ListViewAdapter* con un array de opciones definido en el *strings.xml.*

- Implementa un escuchador sobre la lista, que dependiendo el ítem que sea pulsado lanza las siguientes actividades:
    - **EditProfileActivity**: implementa *AsyncResponse*. Carga la foto de perfil del usuario y los datos del mismo en la vista. Se puede editar todos los datos del usuario y pulsando el botón "guardar" que tiene implementado un escuchador se realiza la petición de *updateUser*. La foto de perfil tiene implementado un escuchador que al pulsar muestra al usuario un diálogo para cambiar la foto mediante:
        - *Galería*: lanza un Intent externo (*MediaStore.ACTION_IMAGE_CAPTURE*).
        - *Cámara*: lanza un Intent externo (*Intent.ACTION_PICK, android.provider.MediaStore.Images.Media.EXTERNAL_CONTENT_URI*).

La respuesta al Intent se procesa en el método *onActivityResult* ya que los *intents* de recuperar la imagen tanto de la cámara como de la galería se inician con *startActivityForResult*. El método *onActivityResult* tiene como parámetros de entrada *int requestCode* (que sirve para saber desde dónde viene llamado el método - Cámara o galería en nuestro caso), *int resultCode* (sirve para saber si se ha realizado correctamente la operación o por el contrario existe algún error) e *Intent data* (que contiene la información de la respuesta).

La información contenida la respuesta del Intent se guardará en un archivo temporal de nuestro dispositivo (modificándolo si existe o creándolo si no existiera), se modifica la foto para que sea de una resolución suficiente y rotándola si es necesario para que se vea correctamente.

Posteriormente se realizará la petición *updateProfileImage* que llevará como parámetros el *bitmap* de la imagen codificada con **Base64** y transformado a *String*.

```
String imgString = Base64.encodeToString(getBytesFromBitmap(bitmap),
      Base64.NO_WRAP);
```

- **ChangePasswordActivity**: implementa *AsyncResponse*. Para actualizar la contraseña se necesita introducir la contraseña actual y la nueva (en 2 ocasiones). Se muestra al usuario una serie de diálogos informativos dependiendo de lo que ha introducido a la hora de cambiar la contraseña (por ejemplo, contraseña actual errónea, las nuevas contraseñas no coinciden, etc.), todos ellos extraídos de *strings.xml*. Realiza la petición de *updatePassword* con los parámetros de las contraseñas codificados con Base64.

```
newPass = Base64.encodeToString(newPassData, Base64.DEFAULT);
```

- **NotificationActivity**: implementa AsyncResponse. Consta de dos selectores para activar/desactivar los avisos por notificaciones push y correo electrónico. Cuando se

activan las notificaciones push lo primero que se hace es registrar contra GCM el dispositivo móvil que nos devuelve el *device_id*, el cual guardaremos a través de la petición *updateDeviceId* en nuestra base de datos.

- **HelpActivity**: muestra un texto informativo. Infla la imagen del logo corporativo.
- **AboutActivity**: muestra un texto informativo. Infla la imagen del logo corporativo.

- Si se pulsa sobre el último ítem de la lista se procederá a cerrar la sesión del usuario mediante la realización de la petición *requestLogout*.

4.1.10 Wearable

Para incluir la funcionalidad del wearable a la aplicación ya existente lo que tenemos que hacer es crear un nuevo modulo de tipo Wear, nosotros lo hemos llamado **intranetwear**. A continuación, asociamos este modulo con nuestra aplicación desde el manifiesto de la siguiente manera:

```
   dependencies {
    wearApp project(':intranetwear')
}
```

Para cambiar la apariencia de las notificaciones en nuestro reloj, lo hacemos desde la clase **MyGcmListenerService** de la aplicación móvil.

Hemos diseñado las vistas del wearable para que tenga varias páginas. La primera con el mensaje que nos llega desde el servidor web, que será idéntico a la notificación push que nos llega al dispositivo:

```
   NotificationCompat.Builder notificationBuilder = new
NotificationCompat.Builder(this)
        .setSmallIcon(R.drawable.aa_logo)
        .setContentTitle(getResources().getString(R.string.app_name))
        .setContentText(message)
        .setAutoCancel(true)
        .setSound(defaultSoundUri)
        .setContentIntent(pendingIntent);
```

Para las siguientes páginas hacemos uso de la clase NotificationCompat.WearableExtender para indeicarle que únicamente se añadiran estas páginas si se está viendo en un reloj inteligente.

```
   NotificationCompat.WearableExtender wearableExtender =
        new NotificationCompat.WearableExtender()
                .setHintHideIcon(true);
```

La segunda para que muestre el cuerpo del mensaje con posibilidad de hacer scroll si el texto es muy grande:

```
   NotificationCompat.BigTextStyle secondPageStyle = new
NotificationCompat.BigTextStyle();
secondPageStyle.setBigContentTitle(subject)
        .bigText(body);
```

```
Notification secondPageNotification =
        new NotificationCompat.Builder(this)
                .setStyle(secondPageStyle)
                .build();
```

Las siguientes páginas son botones que dan funcionalidad a la aplicación. La tercera es para ver el detalle del mensaje y la cuarta para responder al mensaje. Estas acciones vienen lanzadas por **PendingIntents**:

```
//Action 1: Detail view of the message
PendingIntent actionDetailPendingIntent =
     PendingIntent.getActivity(getApplicationContext(), 0, intent,
                                    PendingIntent.FLAG_UPDATE_CURRENT);

   NotificationCompat.Action action1 =
        new NotificationCompat.Action.Builder(R.mipmap.ic_white_email,
                getString(R.string.detail), actionDetailPendingIntent)
                .build();

//Action 2: Response action wearable
Intent replyIntent = new Intent(this, ReplyActivity.class);
PendingIntent actionResponsePendingIntent =
       PendingIntent.getActivity(getApplicationContext(), 0, replyIntent,
                                    PendingIntent.FLAG_UPDATE_CURRENT);
```

Para responder se cargan varias respuestas por defecto desde el strings.xml y la respuesta por voz, que desde la clase **ReplyActivity** extrae la información obtenida por el micrófono y la convierte a texto plano.

```
String replyLabel = getResources().getString(R.string.response);
String[] replyChoices =
getResources().getStringArray(R.array.reply_choices);

RemoteInput remoteInput = new RemoteInput.Builder(EXTRA_VOICE_REPLY)
                            .setLabel(replyLabel)
                            .setChoices(replyChoices)
                            .build();

NotificationCompat.Action action2 =
     new NotificationCompat.Action.Builder(R.mipmap.ic_white_response,
           getString(R.string.response), actionResponsePendingIntent)
                                    .addRemoteInput(remoteInput)
                                    .build();
```

Por último añadimos las páginas a la vista del wearable:

```
   wearableExtender.addPage(secondPageNotification);
wearableExtender.addAction(action1);
wearableExtender.addAction(action2);

notification = notificationBuilder
        .setSmallIcon(R.drawable.aa_logo)
        .setContentTitle(getResources().getString(R.string.app_name))
        .setContentText(message)
        .setAutoCancel(true)
        .setSound(defaultSoundUri)
```

```
        .setContentIntent(pendingIntent)
        .extend(wearableExtender)
        .build();
```

4.2 Vistas y recursos

En este apartado se explicarán los diferentes recursos que se han utilizado para implementar la parte gráfica de la aplicación. Dentro de la carpeta **res/** del proyecto encontramos:

Anim: contiene los archivos.xml que se utilizan para configurar los parámetros de las animaciones cuando realizamos transiciones entre vistas de la aplicación.

- fade_in.xml
- fade_out.xml
- left_in.xml
- left_out.xml
- zoom_forward_in.xml
- zoom_forward_out.xml

Drawable: contiene los archivos.xml que crean los iconos utilizados en la aplicación y archivos *.png* o *.jpg* de imágenes de la aplicación. Está dividido en carpetas que contienen los mismo archivos pero con diferente calidades de densidad de píxeles, y dependiendo del dispositivo dónde se reproduzca la imagen se elegirá una calidad u otra.

- aa_logo.jpg
- button_accept.xml
- button_cancel.xml
- button_delete.xml
- button_login.xml
- fab_label_background.xml
- ic_action_name.png
- ic_add_24dp.xml
- ic_book_24dp.xml
- ic_create_24dp.xml
- ic_description_24dp.xml
- ic_directions_24dp.xml
- ic_edit.png
- ic_email_24dp.xml
- ic_hours.png
- ic_key.png
- ic_location_city_24dp.xml
- ic_map_24dp.xml
- ic_maps.png
- ic_menu_slideshow.xml
- ic_messages.png
- ic_person_24dp.xml

- ic_phone_24dp.xml
- ic_phone_android_24dp.xml
- ic_right.png
- ic_send_24dp.xml
- rounded_button.xml
- side_nav_bar.xml
- user.png

Layout: contiene los ficheros *.xml* correspondientes a las vistas correspondientes a la interfaz gráfica de la aplicación. Los layouts correspondientes a vistas controladas por una Activity siguen la nomenclatura *activity_nombre.xml*, mientras que los que son controlados por un Fragment siguen la nomenclatura *fragment_nombre.xml*.

- activity_about.xml
- activity_add_hour.xml
- activity_add_resource.xml
- activity_change_password.xml
- activity_company.xml
- activity_detail_contact.xml
- activity_edit_hour.xml
- activity_edit_profile.xml
- activity_edit_resource.xml
- activity_help.xml
- activity_main.xml
- activity_message.xml
- activity_navigation_drawer.xml
- activity_new_message.xml
- activity_notification.xml
- activity_resource_group.xml

Menu: contiene la definición de los archivos .xml correspondientes a los menús de la aplicación.

- activity_navigation_drawer_drawer.xml
- detail_email_menu.xml
- home_activity_actions.xml
- hours_menu.xml
- navigation_drawer.xml
- search_menu.xml

Mipmap: contiene los iconos de lanzamiento de la aplicación para las distintas densidades de pantalla existentes al igual que se dividía en la carpeta */drawable*.

Values: contiene los ficheros de recursos de la aplicación como *colors.xml*, *dimens.xml*, *drawables.xml*, *strings.xml* y *styles.xml* con el objetivo de definir los valores y poder acceder a ellos desde cualquier clase o vista de la aplicación.

Un detalle interesante es que el fichero *strings.xml* puede contenerse dentro de otra carpeta denominada *values-x/* dónde la "x" será el descriptor del idioma, los cuales serán cargados por la aplicación dependiendo del idioma de configuración global del dispositivo. De esta manera podemos conseguir traducir nuestra aplicación de manera sencilla a multitud de idiomas.

- colors.xml
- dimens.xml
- drawables.xml
- strings.xml
- styles.xml
- strings.xml (directorio values-en)
- strings.xml (directorio values-es)

Xml: contiene otros ficheros *.xml* utilizados en la aplicación, por ejemplo la estructura de las filas de los *ListViews*.

Assets: contiene los archivos arbitrarios de la aplicación, tales como texto, música o vídeo. Nosotros lo utilizamos para incluir los archivos *Monserrat-Bold.ttf* y *Monserrat-Regular.ttf* para definir la tipografía de los textos de la aplicación.

- fonts
    - Montserrat-Bold.ttf
    - Montserrat-Regular.ttf

4.3 Pruebas

En todo proyecto de software es necesario que el desarrollo tenga una calidad óptima para que sea utilizado por los usuarios sin que falle ni falte ninguna funcionalidad pedida. Para ello se realiza el **aseguramiento de la calidad**, también conocido como **QA** (Quality Assurance).

A lo largo del desarrollo de la aplicación se han ido pasado *pruebas de test* a las funcionalidades implementadas al final de cada Sprint (pruebas incluidas en la *review* de final de Sprint). Las pruebas se han realizado sobre cada Épica del proyecto de la siguiente manera:

Épica 1. Login

- Comprobar diseño.
- Comprobar que la petición de login se hace correctamente.
- Comprobar que la gestión de errores se hace correctamente (usuario o contraseña incorrectos, fallo de conexión, dejar campos vacíos).

Épica 2. Menú Lateral

- Comprobar diseño.

- Comprobar que se realiza correctamente el cambio de vista.

Épica 3. Panel Principal

- Comprobar diseño.
- Comprobar que muestra los últimos 3 registros de cada tabla y que pulsando nos lleva a la vista específica seleccionada.

Épica 4. Registro de Horas

- Comprobar diseño.
- Comprobar la selección del día específico con sus horas si las tiene.
- Comprobar funcionalidad de añadir/editar y su vista.
- Comprobar formulario de imputación (dejando campos vacíos).
- Comprobar popup de eliminación.

Épica 5. Mensajes

- Comprobar diseño.
- Comprobar enviar mensaje.
- Comprobar eliminación de mensaje/s.
- Comprobar búsqueda *barra de búsqueda*.
- Comprobar abrir mensaje.
- Comprobar redactar/responder mensaje.
- Comprobar cambio entre Enviados y Recibidos.

Épica 6. Recursos

- Comprobar diseño.
- Comprobar vista/editar recursos.
- Comprobar añadir recurso dejando campos vacíos.

Épica 7. Contactos

- Comprobar diseño.
- Comprobar listado navegadores del dispositivo.
- Comprobar petición al navegador.
- Comprobar llamada telefónica.
- Comprobar email.

Épica 8. Ajustes / Mi perfil

- Comprobar diseño.
- Comprobar que pulsando en cada botón nos lleva a la vista específica.
- Comprobar formulario editar perfil y cambiar foto del empleado.
- Comprobar notificaciones y la comunicación con Google Cloud Messaging.
- Comprobar Acerca de.
- Comprobar Ayuda.

- Comprobar Logout.

Épica 9. Wearable

- Comprobar diseño.
- Comprobar funcionalidades del wearable.
- Comprobar comunicación entre app-wearable y wearable-app.

Si una prueba no pasa el aseguramiento de la calidad en la *review del Sprint*, se reporta la incidencia con un comentario y se incluye el fallo como nueva *historia de usuario* en el siguiente *Sprint*, con la prioridad de resolución mayor que las otras historias de usuario.

Todas las pruebas han sido finalmente pasadas satisfactoriamente.

Capítulo 5. Conclusions

This section shows the achieved conclusions after the development of the project and the analysis of the objectives. Also includes the possible integrations that can be performed within the application.

The document's introduction exposes the functionality of the mobile application and the main requirements which had to be implemented for the correct ending of the project. By dividing these objectives into several sections we can make a study of the current status:

Realization of the mobile application, the main objective of the project. We can conclude that the mobile application design, the development of functionalities and communication with the server has been carried out successfully. Each application module has been carried out in a satisfactory manner too:

- Principal panel: it displays the latest information of the user and the chance to edit it.
- Resources: it displays the resource groups, the list of resources available to the user and the chance to add, edit or delete a resource.
- Timesheet: it displays a calendar in the top and the list of hours at the bottom. It also fulfils the requirements of add, edit or delete an imputed hour.
- Messages: it displays input and output messages, providing the possibility of creating a new message, reply or delete an existing one. It is also implemented the option to filter any messages with the search bar at the top menu.
- Contacts: it displays the list of companies and the belonging contacts and the detailed information of each one. It allows to open the map, to start navigation to the company, to call the contact phone and to email.
- My profile: it implements options to edit profile picture and user's data, to enable push notifications and to log off. It also shows information concerning the company.

After the analysis of the state of the application, we can conclude that it has well fulfilled with the main objective of this project.

Integration with Android wearable and communication with Google Cloud Messaging, fundamental requirement of the project and one of the most ambitious, because that's the part of research and development. After the study and implementation of the notification service, we can conclude that communication with GCM has been well established and the interaction between the wearable and the application is also optimal.

User experience is a secondary requirement which has been taken into account in the implementation of the project. We can conclude that the navigation in an application is fluid (spending few device's resources) and always kept informed to the user of the processes is carried out in the main thread and in secondary threads.

Personally, after the ending of the project I can conclude that I have learned to work in team better and how to manage projects in software development companies. It has been also possible to carry out a continuous monitoring of the project due to the work methodology used.

This project is a part that includes more content and effort. First, we have the part of the backend, that contains the API server, developed over the years as internal product. Then we have the native applications developed in Android (explained in this document) and iOS (developed by another co-worker of the company).

After analysing the requirements demanded at the beginning of the project and after passing the tests we can conclude that it has been carried out successfully. We can also propose several future lines to the project, explained in the next chapter.

5.1. Future lines

5.1.1 Expansion and Integration of modules

In the first version of the mobile application we have included the most important modules of the Intranet after discussing it with the people in charge of the company, but there exist interesting web server modules which can be integrated into the worker's application.

We can **include these modules** in subsequent versions:

- My Documents: List of relevant user's documents such as payrolls each month.
- My Surveys: list of surveys to determine the status of the projects (current status of the project, blocking tasks) which the worker must fill to have a better management control of projects in the company.
- Material and feed: module that gives the option of ordering office supplies or any consumable to office's administrator.

In this project we have focused on the part of worker's Intranet, but there is also a part of **administrator's Intranet**, which includes the module Commercial, Users (to add or delete users, manage permissions) and Human Resources (management of selection processes). Therefore, another future line would be the develop of the administrator's part to be able to access from the mobile application.

The application is currently translated into two languages (Spanish and English). We could create more translation files to **internationalize the application**.

Finally, **notification's management** focuses only on the messages module, but it could be implemented in more application modules, such as to notify that user has a new resource available, new customer available or new project available.

5.1.2 Code refactor

We have tried to keep a proper organization and structure of the code, by dividing the classes. The project has been developed for the 4.4 target (KitKat), but the trend of the market will be an expansion of Android devices with Android 5.0 or higher, to the detriment of the devices currently using version 4.4. Therefore, one of the next steps would be to try to refactor the code to optimize development for version 5.0 of Android.

Bibliografía

[1] The Scrum Theory, The Scrum Guide. En: *Scrum Guides* [en línea]. Disponible en: http://www.scrumguides.org/scrum-guide.html [consulta: 12 mayo 2016]

[2] Deloitte. "Global Mobile Consumer Survey, 2015", *deloitte.com*, (Noviembre 2015) [en línea]. Disponible en: http://www2.deloitte.com/es/es/pages/technology-media-and-telecommunications/articles/Consumo-movil-2015.html [Consulta: 04 de junio 2016]

[3] Android, *android.com* [en línea]. Disponible en: https://www.android.com/intl/es_es/ [Consulta: 05 de junio 2016]

[4] iOS, *apple.com* [en línea]. Disponible en: http://www.apple.com/es/ios/ [Consulta: 05 de junio 2016]

[5] Windows Phone, *microsoft.com* [en línea]. Disponible en: https://www.microsoft.com/es-es/windows/phones [Consulta: 05 de junio 2016]

[6] Apache Cordova, *cordova.apache.org* [en línea]. Disponible en: https://cordova.apache.org/ [Consulta: 05 de junio 2016]

[7] AngularJs, angularjs.*org* [en línea]. Disponible en: https://angularjs.org/ [Consulta: 05 de junio 2016]

[8] Ionic, ionicframework.*com* [en línea]. Disponible en: http://ionicframework.com/ [Consulta: 05 de junio 2016]

[9] Google Play. "Aplicaciones", *play.google.com* [en línea]. Disponible en: https://play.google.com/store/apps/details?id=net.artvisual.controlPanelMovil [Consulta: 11 de marzo 2016]

[10] Google Play. "Aplicaciones", *play.google.com* [en línea]. Disponible en: https://play.google.com/store/apps/details?id=com.protectfive.intratime [Consulta: 11 de marzo 2016]

[11] Google Play. "Aplicaciones", *play.google.com* [en línea]. Disponible en: https://play.google.com/store/apps/details?id=personal.andreabasso.clearfocus [Consulta: 11 de marzo 2016]

[12] Google Play. "Aplicaciones", *play.google.com* [en línea]. Disponible en: https://play.google.com/store/apps/details?id=com.insightly.droid&feature=nav_result#?t=W251bGwsMSwyLDNd [Consulta: 11 de marzo 2016]

[13] Google Play. "Aplicaciones", play.google.com [en línea]. Disponible en: https://play.google.com/store/apps/details?id=com.hipchat [Consulta: 11 de marzo 2016]

[14] Google Play. "Aplicaciones", *play.google.com* [en línea]. Disponible en: https://play.google.com/store/apps/details?id=com.yammer.v1&hl=es [Consulta: 11 de marzo 2016]

[15] Google Play. "Aplicaciones", *play.google.com* [en línea]. Disponible en: https://play.google.com/store/apps/details?id=com.lastpass.lpandroid&hl=en [Consulta: 11 de marzo 2016]

[16] Developer Android. “Up and running with material design", *developer.android.com* [en línea]. Disponible en: https://developer.android.com/design/index.html [Consulta: 10 de abril 2016]

[17] Developer Android. “Transmitting Network Data Using Volley", *developer.android.com* [en línea]. Disponible en: https://developer.android.com/training/volley/index.html [Consulta: 20 de mayo 2016]

Anexo I. Abstract

First, we are going to introduce the context of the work, concretely the company's description and the product's description.

Armadillo Amarillo S.C. is a software development company born in 2013 which develop web and mobile. This company has developed since years an intranet as an internal product of the company based on modules that concentrate all the basic managements of itself.

An intranet is a private network accessible only to the company members. It is used to share confidential company information between its members.

Armadillo Intranet is the name of the web service and it is already implemented and in use in the company. The idea of this product was to optimize the need of the company's management. It is also created to offer and market this customizable product to other companies of the TIC sector.

It is divided in two sections depending of the roles of the user, both of them divided in several modules:

Worker intranet.

The functionality of this part is to monitor the activity of the members of the enterprise, have a control of the productive hours and help in communication among workers. It is divided in several modules, in particular, "resources", "timesheet", "messages", "contacts", "surveys", "documents" and "supplies and food".

Administrator intranet.

The functionality of this part is to manage all the members and resources of the company. It includes several modules, such as "commercial", "users", "resources" and "human resources". It also has the correspondent modules of the worker intranet.

This was what the company had implemented when the idea of the project explained in this report was born. This idea arises because of the continuous innovation and evolution of the technology, specifically, mobile technology.

Mobile technology is changing very quickly the behaviour and habits of the world society and have created the dependence and need of being able to access the information immediately.

The goal of developing this application is facilitating access to the web server and increase its use. Thus, the workers could do their daily task in a faster way and the productivity of the company will be increased.

Armadillo Amarillo decided to generate two mobile applications developed in Android and iOS (native develop). The Android application is the one which I develop and explain in this document and the iOS one which has been developed by other member of the company as a TFG as well.

The main objective of this project is to develop a mobile application for managing a company's intranet in Android technology. The project which is exposed in this report is the result of several months of development in the company.

Android is a mobile operating system developed by Google, based on the Linux kernel. An Android application is based on components which are Activities, Services, Broadcast Receivers and Content Providers. The application will be composed of one or more of these components. They have to be declared in AndroidManifest.xml which is a file that contains metadata for a group of accompanying files.

We have included in this project a section of push notifications and we have used Google Cloud Messaging (GCM) to manage this information. GCM is a mobile service developed by Google that allows application developers to send push notifications with information on their servers. We will explain the communication between server-GCM and GCM-device in the relevant section of the document.

The IDE selected to develop this project is Android Studio which is the official IDE for Android developing. This IDE is based on IntelliJ.

Before start developing we made a comprehensive research of the market to see the different mobile applications which have similar functionalities to ours. We divided this research in 4 different subparts because there is no application that attach all this features at the same time.

- Timekeeping, where we can stand out Sesame, Intratime and ClearFocus.
- Clients management, where we can stand out Insightly.
- Communication between workers, where we can stand out Hipchat and Yammer.
- Resource management, where we can stand out LastPass.

Once the market research was done we can start developing our application. First, we are going to explain the functionalities of our product but first of all, we have to clarify that the company had created the web service and the backend before this project had started. Now, we are going to explain what features we have implemented in the project:

API.

It has been necessary to implement various API services with the objective that the mobile application can obtain the information from the backend. All the request made to the API follows an JSON structure compound of three keys (service, authorization and arguments) with its corresponding values. We will explain it with more details in the relevant section of the document.

Android Mobile Application.

The mobile application will be an adapted version of the web service. The first view of the application is the Login layout, where the user can introduce the credentials. It is contemplated an error management to let the user know the error detail if something goes wrong.

When the user goes into the application (correct login) it shows a lateral menu which contains the main views of the application. We can find the Principal Panel, Resources, Timesheet, Messages, Contacts and Settings/My profile. Now, we are going to explain the different modules of the the lateral menu.

Resources module: it will show a list with the group's resources available to the user. Then, the user can access to specific resources of the group by clicking over one of the list. There is an option to add a new resource and see the detail of a specific resource by clicking over it, where you can edit or delete it.

Timesheet module: it will show a material design calendar on the top of the view and a list of imputed hours in the bottom. You can add a new hour by clicking over de FAB and see the detail of one hour by clicking over it, where you can edit or deleted it too.

Messages module: it will show the inbox and outbox of the user. At the top of the view it shows a search bar to facilitate to find some email filtering by user, email, title or body. There is an option to write a new email by clicking the FAB and see the detail of one message by clicking over it. In the detail's layout you can reply or delete the message by clicking in the reply or delete button in the top of the screen.

Contacts module: it will show a list of our clients companies and we can access to the list of company's contacts by clicking over it. We can start the navigation or open the location with Google Maps by clicking the specific FAB. To see the detail of the contact (phone number, email, workstation, etc.) we have to click in one item of the list. Once inside the detail view we can make a phone call or write an email directly by clicking over the option we want.

My profile module: it will show the profile image of the user and the settings. These settings are:

- Edit profile: where you can edit your personal info and your profile image.
- Change password.

- Notification settings: where you can activate/deactivate the push and your email notifications.
- Help and about us: which shows company's information.
- Logout: which close the user session.

Principal panel module: it will show the last three resources, messages and imputed hours for the user with the possibility to see the detail of each one and an direct access to create a new one.

Wearable application.

The push notification will be reproduced in the wearable device with a preview of the title and body. In the wearable, the user can see the notification and swipe to access various functionalities from the watch, such as see the message detail in the device, reply with predefined text or voice command.

To make sure that all the features has been well implemented all of them will have to pass a QA test descripted in the document. We can conclude that all the functionalities have pass the QA as is shown in the relevant section of this document.

To have a continuous monitoring of the project and to make sure that we will reach its delivery we have follow the SCRUM methodology. This agile methodology is based on an incremental development overlapping phases instead of performing one after another in a sequential cycle.

The project is divided in phases which is denominated "Sprints" with a 2-week duration. The development of the project was divided in tasks called "Epics" which were divided in tasks of less time called "User stories". This user stories are included in the different Sprint along the development of the product.

We used JIRA project manager. This manager has a board that consists of several columns with the corresponding tasks of the current Sprint, such as "To do", "In progress", "Blocked", "QA" and "Done". We also used a version control system client called SourceTree based on Git.

Finally, to conclude this abstract we are going to check if the requirements has been successfully done. For this purpose, we will review the objectives of this project.

First, the main objective of this project was the **realization of the mobile application**. We can conclude that the development of functionalities and communication with the server has been carried out successfully.

Second, the **integration with the Android wearable** works fine and the communication between the wearable and device works correctly too.

Third, the communication between our server and **Google Cloud Messaging** and between Google Cloud Messaging and operative devices had been settled down correctly.

Finally, we can expose that the navigation in the mobile application is fluid and the charging times doing requests to the server are fast. So the objective of get a good user experience is completed too.

Keywords

Android, AsyncTask, Intranet, API, CGM, AVD, SDK.

Anexo II. Traducción al castellano del capítulo "Introducción"

En este capítulo se realizará un breve resumen de los ámbitos más importantes del proyecto, incluyendo los motivos por los que se realiza, el alcance y la metodología que se ha seguido para realizar todas las fases. Al final de este capítulo se incluye la estructura que compone la memoria.

1.1 Introducción del proyecto

El objetivo principal del proyecto es desarrollar una **aplicación móvil para la gestión de la intranet de una empresa**. El público objetivo es cualquier empresa activa del mercado y los usuarios finales, sus empleados. Este primer prototipo está pensado para PYMES, aunque es un proyecto escalable a largo plazo.

Armadillo Amarillo S.C., empresa dónde se ha realizado este proyecto, es una empresa dedicada al mundo de la tecnología, con una amplia experiencia en el desarrollo de software y la implantación de servicios.

Armadillo Intranet es un producto interno de la empresa Armadillo Amarillo utilizado para gestionar la intranet de la compañía. Este producto ha sido desarrollado con la intención de ser vendido a otras compañías del sector para facilitar el manejo de su propia intranet.

Armadillo Intranet ha sido el resultado de la colaboración de varios miembros de la empresa trabajando en equipo. La empresa tenía desarrollado un servicio web compuesto por módulos y su *backend* correspondiente. Lo que se quiere conseguir con este proyecto es llevar el servicio web a tecnología móvil, concretamente *Android* e *iOS*.

Como parte activa del equipo, mi trabajo ha consistido en implementar la aplicación móvil en tecnología **Android** con la integración de **Android Wear** y el desarrollo de varios **servicios de la API**, la cual no estaba desarrollada al existir únicamente el servidor web.

Los pasos que se han llevado a cabo para la completa realización del proyecto han sido:

- Estudio del sistema de partida de la empresa y compresión de su funcionamiento.
- Análisis de la ampliación del sistema de partida para que sirva como backend de las aplicaciones móviles a desarrollar.
- Diseño y desarrollo de los servicios del backend.
- Diseño y desarrollo de la aplicación móvil en tecnología Android.
- Diseño y desarrollo de la aplicación wearable de Android.
- Realización de la documentación.

El objetivo principal de este proyecto es cubrir varias necesidades relacionadas con el mundo empresarial. Primero, agilizar el acceso a los datos del día a día de una empresa (recursos tales como

accesos, contraseñas, documentos, etc.) y a los datos de los clientes (teléfono de contacto, email, dirección física de la empresa, etc.). Por otro lado, facilitar la comunicación entre los empleados y el control de la gestión de los proyectos mediante la imputación de horas.

Existe otra aplicación móvil para *iOS* desarrollada en paralelo a la que se detalla en este documento, también realizada como parte de otro TFG en esta compañía.

1.2 Motivación

En España, el porcentaje de población adulta que utiliza un *smartphone* a diario para conectarse a Internet es muy alto, y cada año va creciendo más. La población mundial se está adaptando rápidamente a las nuevas tecnologías y el mercado *mobile* es muy importante dentro de este sector.

Dado que el *smartphone* siempre va contigo, la rapidez con la que puedes acceder al dispositivo móvil es mucho mayor que al ordenador, siendo muchas veces los mismos servicios los que encuentras en ambos medios.

Hoy en día existen muchas aplicaciones móviles para facilitar la vida diaria de una empresa (para comunicación de empleados, para el control de horas de trabajo, etc.). Lo que se pretende es unificar todas esas funcionalidades en una sola aplicación, para desperdiciar el mínimo tiempo posible y ser mucho más eficiente en el trabajo.

Hay que tener en cuenta el valor añadido que se le da a estos servicios, que es la movilidad, y el fácil acceso en todo momento, con lo que se pretende conseguir un aumento en el uso de los servicios.

1.3 Alcance del proyecto

Se propone el desarrollo de una aplicación móvil realizada en tecnología *Android* y la integración con el *wearable* junto con la implementación de los servicios de la API necesarios para obtener los datos con los que dotar a la aplicación de sentido, realizado en tecnología PHP.

Alcance de la aplicación:

- **Panel Principal**: la vista mostrará los últimos 3 recursos disponibles para el usuario, los 3 últimos mensajes recibidos y las 3 últimas horas imputadas.
- **Recursos**: la vista mostrará los distintos grupos de recursos disponibles para el usuario.
- **Registro de horas**: la vista mostrará un calendario con el mes actual y un listado de horas imputadas para el día seleccionado.
- **Mensajes**: la vista mostrará la bandeja de entrada y de salida del usuario y una barra de búsqueda para poder filtrar los mensajes.

- **Contactos**: la vista mostrará un listado de empresas que se tengan como clientes con la opción de ver el detalle de contacto.

Alcance del wearable:

- **Notificaciones**: se mostrará una notificación de mensaje nuevo con la posibilidad de ver el detalle desde el dispositivo, responder con texto predefinido y por comando de voz.

Alcance de la API:

- **Servicios**: se implementarán las funciones necesarias para posibilitar la comunicación del dispositivo móvil con el backend.
- **GCM**: peticiones a *Google Cloud Messaging* necesarias para la comunicación y habilitación de notificaciones para el dispositivo.

1.4 Metodología y planificación

La metodología elegida para llevar a cabo el proyecto es **SCRUM** [1], método de desarrollo ágil de software que divide las etapas de desarrollo en diferentes Sprints, en nuestro caso de una duración de 2 semanas, con una revisión a final de cada uno para ver los avances y el estado del proyecto.

El equipo de trabajo se divide en:

- **Equipo de desarrollo**: responsables de entregar el producto a final de cada Sprint. Las personas que componen el equipo (desarrolladores de Android, iOS y backend) se reparten las tareas a realizar de la manera más óptima para el desarrollo del producto.
- **Product Owner**: persona que representa al cliente, en nuestro caso, al ser un producto interno de la empresa, representa a la propia empresa.
- **Scrum Master**: persona encargada de facilitar el desarrollo de la metodología y de garantizar que el producto llega a los objetivos marcados. No gestiona el equipo de desarrollo, ya que este se organiza a sí mismo. En este proyecto este papel ha sido llevado a cabo por el tutor de la empresa.

El calendario propuesto se divide en:

- Sprint 0: 1 febrero – 9 febrero
- Sprint 1: 10 febrero – 24 febrero (review 25 febrero)
- Sprint 2: 25 febrero – 10 marzo (review 14 marzo)
- Sprint 3: 14 marzo – 27 marzo (review 28 marzo)
- Sprint 4: 28 marzo – 11 abril (review 12 abril)
- Sprint 5: 12 abril – 26 abril (review 27 abril)

- Sprint 6: 27 abril – 11 mayo (review 12 mayo)

1.5 Fases de trabajo

El proyecto se ha dividido en varias fases. Primero una fase de conceptualización, diseño de la aplicación y estimación de tiempos de desarrollo que se llevó a cabo durante el Sprint 0. Después se procedió a la fase de desarrollo y pruebas, la más larga en términos de tiempo, que se llevó a cabo durante los Sprints 1, 2, 3 4 y 5. Por último, la redacción de la documentación se realizó durante el Sprint 6.

1.6 Medios empleados

A continuación listaremos el hardware y software empleados para el desarrollo de este proyecto junto con sus características técnicas.

Hardware:

- MacBook Pro (13 pulgadas, principios de 2011): ordenador personal de desarrollo.
  - Procesador: 2,3 GHz Intel Core i5
  - Memoria: 8 GB 1600 MHz DDR3
  - Gráficos: Intel HD Graphics 3000 512 MB

- Dispositivo Samsung Galaxy Core 2: smartphone para desarrollo y pruebas.
  - Procesador: 1.2GHz Quad-Core
  - Pantalla: 4.5" (114.2mm)
  - Versión Android: 4.4.2 (KitKat)

- LG Watch Urbane: reloj inteligente para desarrollo y pruebas.
  - Sistema Operativo: Android Wear
  - Pantalla táctil circular de 1.3"
  - Compatible con Android 4.3 o superior

Software:

- Android Studio 1.5.1: IDE de desarrollo para la aplicación Android.
- IntelliJ IDEA 15: IDE de desarrollo para los servicios PHP del backend.
- Postman: cliente web para realizar peticiones a la API.
- Emulador Android Wear Round API 21: software para emular el wearable en el ordenador.
- Genymotion: software para emular el smartphone en el ordenador.

1.7 Entorno socio-económico

Según el informe **Consumo Móvil 2015** elaborada por **Deloitte** [2], España se sitúa a la cabeza a nivel mundial en penetración de smartphones con un **88%** de la población (en Europa un 82%), incrementando 3 puntos porcentuales en un año y 19 puntos con respecto 2013. Este informe también revela que el 50% de los usuarios móviles consultan su dispositivo en los primeros y últimos minutos del día.

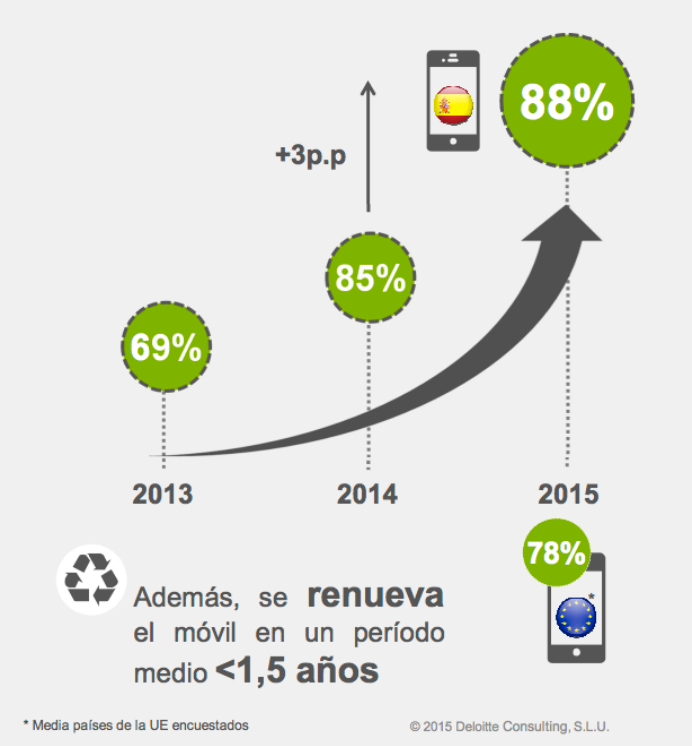

Ilustración 51 – Penetración Smartphone en el mercado español

La tecnología móvil está cambiando el comportamiento y hábitos de la sociedad mundial de manera muy rápida, tanto en mercados estables y avanzados como en mercados emergentes de países en desarrollo.

A lo largo de los últimos años las aplicaciones móviles han revolucionado la forma de vivir de la sociedad global. La mayoría de las empresas contemplan realizar una integración de su negocio con la telefonía móvil si es que no la han desarrollado ya, lo que genera un impacto económico muy fuerte en el sector TIC (Tecnologías de la información y la comunicación).

El éxito que han tenido las aplicaciones móviles en la sociedad es debido a la labor facilitadora que implantan en los usuarios, que pueden acceder a la información en cualquier lugar y momento desde el menú de su dispositivo.

Esta tendencia ha generado en el consumidor la necesidad de tener aplicaciones para todas las funcionalidades de su vida cotidiana. Si quieres saber dónde ir a cenar esta noche buscas en alguna

aplicación móvil los restaurantes cerca de tu ubicación, si quieres coger un taxi lo pides a través de la aplicación, si quieres escuchar música lo haces a través de una aplicación, si quieres consultar tus cuentas de tu entidad bancaria lo haces a través de su aplicación, etc., sin olvidarnos del fuerte impacto que generan las aplicaciones enfocadas a la comunicación entre personas y redes sociales.

Las aplicaciones móviles han generado la dependencia en la sociedad de que sea necesario poder acceder a la información sin perder tiempo práctico en el camino.

Pero no solo se centran en facilitar el día a día de la sociedad en labores más livianas, también existe un nicho de mercado importante orientado a la tecnología móvil para la salud, que puede mejorar notablemente la vida de la sociedad.

En definitiva, lo que se pretende con este proyecto es facilitar la vida de los trabajadores de cualquier empresa para que su tiempo de trabajo sea lo más eficiente posible, incrementando el tiempo de productividad real del trabajador y así generar un impacto económico mayor en la empresa.

1.8 Marco Regulador: Política de privacidad y tratamiento de datos

En la aplicación móvil explicada en este documento es necesario que los usuarios introduzcan información personal como nombre, apellidos, NIF, correo electrónico, teléfono de contacto, etc. y es por ello que se le aplica la ley orgánica de protección de datos de carácter personal.

Esta ley *"[...] tiene por objeto garantizar y proteger, en lo que concierne al tratamiento de los datos personales, las libertades públicas y los derechos fundamentales de las personas físicas, y especialmente de su honor e intimidad personal y familiar.*

Esta ley afecta a todos los datos que hacen referencia a personas físicas registradas sobre cualquier soporte, informático o no. [17]*[...]"*

También se hace uso de la ley General de las Telecomunicaciones. En concreto, podemos centrarnos en el Artículo 41. Protección de los datos de carácter personal, que incluye *"[...] la garantía de que sólo el personal autorizado tenga acceso a los datos personales para fines autorizados por la Ley, la protección de los datos personales almacenados o transmitidos de la destrucción accidental o ilícita, la pérdida o alteración accidentales o el almacenamiento, tratamiento, acceso o revelación no autorizados o ilícitos y la garantía de la aplicación efectiva de una política de seguridad con respecto al tratamiento de datos personales* [18] *[…]"*. También se hace

[17] La Ley Orgánica 15/1999 de 13 de diciembre de Protección de Datos de Carácter Personal (BOE núm. 298 de 14 de Diciembre de 1999)

[18] Ley 32/2003, de 3 de noviembre, General de Telecomunicaciones. (BOE núm. 264 de 04 de Noviembre de 2003)

uso del Artículo 43. Cifrado en las redes y servicios de comunicaciones electrónicas, que indica que *"[...] cualquier tipo de información que se transmita por redes de comunicaciones electrónicas podrá ser protegida mediante procedimientos de cifrado [...]".*

Por tanto, los datos manejados en esta aplicación serán almacenados en una base de datos de manera automatizada responsabilidad de **Armadillo Amarillo S.C.**

La información recogida puede incluir: *datos generales de contacto* (nombre, apellidos, dirección, teléfono de contacto, email …) y *datos particulares* (información laboral).

Armadillo Amarillo se compromete a garantizar que la información gestionada solo será accesible por personas autorizadas, y su circulación se limitará a las acciones permitidas por la empresa contratante.

Se tratarán los datos de manera confidencial, tanto los datos correspondientes a los empleados de la empresa contratante como los datos de los contactos comerciales de la misma, quienes serán informados también del tratamiento al que serán sometidos sus datos.

En términos técnicos, la normativa vigente solo afectará a la hora de proteger los datos de los usuarios por motivos de seguridad. Es por ello que se ha codificado la comunicación entre la aplicación móvil y el servidor mediante la codificación en Base64. Este es un sistema de numeración posicional que usa 64 como base. Es la mayor potencia de dos que puede ser representada usando únicamente los caracteres imprimibles de ASCII. Por otro lado, la transmisión de datos entre la aplicación y el servidor se realizará sobre HTTPS (Hypertext Transfer Protocol Secure).

1.9 Estructura del documento

En el capítulo 1, **Introducción**, se realizará un breve resumen del documento. Se analizará el alcance del proyecto, las motivaciones para llevarlo a cabo, la metodología empleada y un breve desglose de las fase del proyecto.

En el capítulo 2, **Estado del arte**, se analizará el estado de las tecnologías móviles hasta el momento, haciendo más hincapié en el entorno de desarrollo elegido.

En el capítulo 3, **Especificación de requisitos y descripción de la aplicación**, se realizará un análisis de las funcionalidades de la aplicación y se detallarán los requisitos que se deben cumplir. Se analizará la metodología de trabajo utilizada y la estimación del proyecto. También se pondrá en contexto del estado en el que nos encontramos (aplicación web) antes de comenzar el desarrollo.

En el capítulo 4, **Desarrollo de la aplicación**, se realizará un análisis técnico del desarrollo de la aplicación enfocándonos en el diagrama y estructura de clases. También se analizará las pruebas que se han llevado a cabo para determinar la calidad del software.

En el capítulo 5, **Conclusiones**, se presentarán las conclusiones extraídas tras realizar el proyecto y las posibles líneas futuras de desarrollo.

Anexo III. Traducción al castellano del capítulo "Conclusiones"

En este apartado se muestran las conclusiones extraídas tras el desarrollo del proyecto y el análisis de los objetivos del mismo. Se incluyen también las posibles integraciones que se pueden llevar a cabo dentro de la aplicación.

En la introducción a este documento se exponía la funcionalidad de la aplicación móvil y los requisitos principales que debían ser implementados para la correcta finalización del proyecto. Dividiendo estos objetivos en varios apartados podemos hacer un estudio de su estado actual:

Realización de la aplicación móvil, siendo este el objetivo principal del proyecto. Podemos concluir que se ha llevado a cabo satisfactoriamente el diseño de la aplicación móvil, el desarrollo de las funcionalidades y la comunicación con el servidor. Cada módulo de la aplicación se ha llevado a cabo de manera satisfactoria:

- Panel Principal: muestra la información más reciente del usuario y la posibilidad de editarla.

- Recursos: muestra los grupos de recursos, el listado de recursos disponibles para el usuario y la posibilidad de añadir, editar o borrar un recurso.

- Registro de horas: muestra un calendario en la parte superior y el listado de horas en la inferior. También cumple con los requisitos de añadir, editar o borrar una hora imputada.

- Mensajes: muestra la bandeja de entrada y salida de los mensajes, brindando la posibilidad de crear un nuevo mensajes, responder o borrar uno existente. También se implementa la opción de filtrar cualquier mensajes con la barra de búsqueda en el menú superior.

- Contactos: muestra el listado de empresas y los contactos pertenecientes a la misma con la información detallada de cada uno. Permite abrir el mapa, iniciar navegación a la empresa, llamar al teléfono de contacto y enviar un correo electrónico.

- Mi perfil: implementa las opciones de editar los datos y foto de perfil, activar notificaciones push y cerrar sesión. Muestra también información de ayuda referente a la empresa.

Tras el breve análisis del estado de la aplicación podemos concluir que se ha cumplido de manera satisfactoria con el objetivo principal de este proyecto.

Integración del wearable y comunicación con Google Cloud Messaging, siendo este un requisito fundamental del proyecto y uno de los objetivos más ambiciosos del mismo, ya que supone la parte

de investigación y desarrollo. Tras el estudio e implantación del servicio de notificaciones podemos concluir que la comunicación con GCM se ha establecido de manera satisfactoria y la interacción del wearable con la funcionalidad de la aplicación también es óptima.

La **experiencia de usuario** es un requisito secundario que se ha tenido en cuenta en la realización del proyecto. Podemos decir que la navegación en la aplicación es muy fluida, gastando pocos recursos del dispositivo y manteniendo siempre informado al usuario de los procesos que está llevando a cabo tanto en el hilo principal como en hilos secundarios.

Personalmente, tras la realización del proyecto puedo concluir que he aprendido a trabajar mejor en equipo y cómo se gestionan los proyectos en empresas de desarrollo software. Mediante la metodología de trabajo empleada se ha podido llevar a cabo un seguimiento continuado del proyecto consiguiendo reorganizar las tareas y tiempos que se debían emplear en cada fase de desarrollo.

Este proyecto es una parte que engloba más contenido y esfuerzo. Por un lado tenemos la parte del backend, que contiene la API del servidor, desarrollado como producto interno de la empresa durante años y por el otro tenemos las aplicaciones nativas desarrolladas en Android (explicada en este documento) y en iOS (desarrollada por otro compañero de la empresa).

Tras el análisis de los requisitos que se pedían al principio del proyecto y el posterior cumplimiento de las pruebas de QA podemos ver que se han llevado a cabo todas satisfactoriamente. Aún así, al tratarse de un proyecto escalable, podemos abrir la puerta a varias líneas futuras y mejoras.

5.1 Líneas futuras

5.1.1 Ampliación/Integración de módulos

En esta primera versión del proyecto se han incluido los módulos más importantes de la Intranet tras discutirlo con los responsables de la empresa, pero cómo hemos explicado antes en el servidor web se incluyen más módulos interesantes para ser integrados en la aplicación del trabajador.

En próximas versiones se pueden **incluir los módulos**:

- **Mis documentos**: listado de los documentos de importancia para el usuario como son las nóminas de cada mes.
- **Mis encuestas**: listado con diferentes encuestas para conocer el estado de los proyectos (situación actual del proyecto, tareas bloqueantes, etc.) que el trabajador debe rellenar de forma periódica para tener un mejor control de la gestión de los proyectos por parte de la empresa.
- **Material y alimentos**: módulo que da la opción de pedir material de oficina o algún consumible al administrador de la oficina.

En este proyecto nos hemos enfocado en la parte de la Intranet del trabajador, pero también existe la parte de **administración de la Intranet**, que incluye los módulos de *Comercial*, *Usuarios* (para dar de alta o baja usuarios, gestionar permisos, etc.) y *Selección y RRHH* (gestionar procesos de selección y ofertas de empleo). Por tanto, otra línea de futuro **sería diseñar la parte de administración** de la Intranet para poder acceder desde la aplicación móvil.

La aplicación actualmente se encuentra traducida a dos idiomas (español e inglés). Se podría crear más archivos de traducción para **traducir la aplicación a más idiomas**.

Por último, la **gestión de la información de las notificaciones** se centra únicamente en el módulo de mensajes, pero se pueden implantar en más módulos de la aplicación, por ejemplo para notificar al usuario que tiene un nuevo recurso disponible, nuevo cliente disponible o nuevo proyecto disponible.

5.1.2 Refactorización del código

Durante este proyecto se ha intentado llevar una organización y estructura correcta del código, mediante la división de las clases. El proyecto se ha desarrollado para el target de 4.4 (KitKat), con el avance del tiempo, la tendencia del mercado Android será ampliar los dispositivos con Android 5.0 o mayor, en detrimento de los dispositivos que actualmente utilizan la versión 4.4. Por tanto, uno de los siguientes pasos sería tratar de refactorizar el código para **optimizar el desarrollo para la versión 5.0** de Android.

Anexo IV. Terminología

Glosario de términos

Android. Sistema operativo basado en el núcleo Linux. Desarrollado inicialmente por Android Inc.

Android Studio. Entorno de desarrollo integrado para la plataforma Android que sustituyó a Eclipse como IDE oficial.

Gradle. Herramienta flexible que se utiliza para automatizar la construcción de los proyectos desarrollados en Android. Automatiza las tareas de compilación, testing y empaquetado.

Android Manifest. Archivo de configuración donde podemos se aplican las configuraciones básicas de nuestra aplicación. Se declaran los componentes del proyecto, los permisos que se utilizan y las librerías externas utilizadas.

IntelliJ IDEA 15. Entorno de desarrollo integrado para el desarrollo de software. Desarrollado por JetBrains.

Postman. Es un cliente REST que se ejecuta como una aplicación en el navegador Chrome. Se utiliza para la interconexión con APIs REST.

REST (representational state transfer). Estilo de arquitectura de software de la World Wide Web. El propósito de la arquitectura REST es inducir el rendimiento, la escalabilidad, la sencillez, la capacidad de modificación, la visibilidad, portabilidad y fiabilidad.

Emulador. Software que permite ejecutar programas en una plataforma (sea una arquitectura de hardware o un sistema operativo) diferente de aquella para la cual fueron desarrollados originalmente.

Genymotion. Emulador de Android de la compañía francesa de software del mismo nombre. Desarrollado en el software de virtualización VirtualBox, emulador basado en Oracle.

Layout. Vista de la aplicación.

JIRA. Aplicación basada en web para el seguimiento de errores, de incidentes y para la gestión operativa de proyectos.

SCRUM. Metodología ágil de trabajo basada en Sprints.

Sprint. Período en el cual se lleva a cabo el trabajo en sí. Es recomendado que la duración de los sprints sea constante y definida por el equipo basándose en su propia experiencia, normalmente 2 o 3 semanas.

Modelo-Vista-Controlador (MVC). Arquitectura de software que separa los datos y la lógica de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

AsyncTask. Objeto Java que permite realizar operaciones en segundo plano mediante otro hilo de ejecución que no sea el principal.

Tecnología push. Forma de comunicación en la que la petición de envío tiene origen en el servidor, por oposición a la tecnología pull, en la que la petición tiene origen en el cliente.

Google Maps. Servidor de aplicaciones de mapas en la web que pertenece a Alphabet Inc.

Intranet. Red informática que utiliza la tecnología del Protocolo de Internet (IP) para compartir información, sistemas operativos o servicios de computación dentro de una organización.

Técnica Pomodoro. Método para mejorar la administración del tiempo. Esta técnica usa un reloj para dividir el tiempo dedicado a un trabajo en intervalos de 25 minutos, *pomodoros*, separados por pausas.

Swipe. Gesto lateral que se realiza con el dedo en un dispositivo táctil.

Abreviaturas

API (Application Programming Interface). Conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

CGM (Google Cloud Messaging). Servicio móvil desarrollado por Google que permite transmitir datos desde los servidores a una aplicación cliente mediante tecnología push.

APK (Android Application Package). Es el paquete para el sistema operativo Android que sirve para instalar componentes empaquetados.

IDE (Integrated Development Environment). Aplicación informática que proporciona servicios integrales para facilitar el trabajo a los desarrolladores la implementación de código en un cierto lenguaje.

AVD (Android Virtual Device). Permite definir las características del teléfono, Tablet, Android Wear o Android TV que se desea simular en el emulador.

SDK (Software Development Kit). Conjunto de herramientas de desarrollo de software que le permite al desarrollador de software crear aplicaciones para un sistema concreto.

NDK (Native Development Kit). Conjunto de herramientas que permite implementar partes de la aplicación utilizando lenguajes de código nativo, como C y C ++.

FAB (Floating Action Button). Botón con forma de círculo que se muestra flotando por encima de la interfaz de usuario de la aplicación.

GPS (Global Positioning System). Sistema de navegación basado en el espacio que proporciona información sobre la posición y la hora.

Lo primero, establecemos la conexión entre el *dispositivo* y el *wearable*. Esto se puede hacer de dos maneras dependiendo si estamos realizando la conexión *dispositivo-emulador* o *dispositivo-wearable*.

Para la conexión con el emulador iniciado con *Android Studio* se tiene que introducir por el terminal el siguiente comando:

```
adb -d forward tcp:5601 tcp:5601
```

En el dispositivo tendremos que indicarle que la conexión queremos realizarla con el emulador.

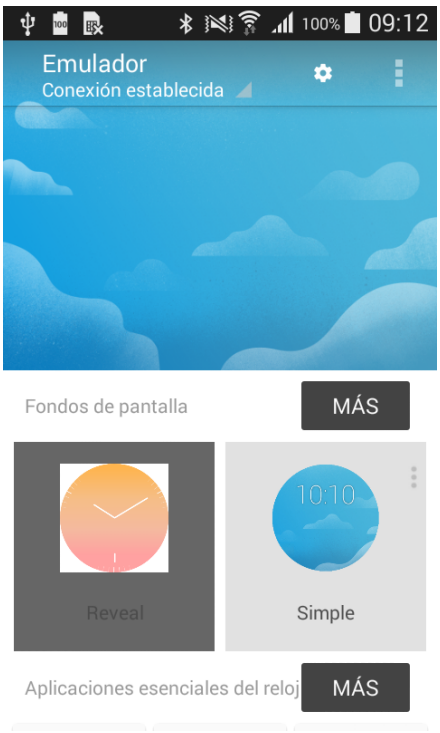

Ilustración 52 – Conexión exitosa entre dispositivo y emulador.

Para la conexión entre dispositivo y wearable reales, la conexión es mucho más sencilla y directa mediante *bluetooth*.

Anexo VI. Configuración entorno con GCM

En este anexo explicaremos como se establece conexión con **Google Cloud Messaging** y la gestión de las notificaciones push desde nuestra aplicación.

Lo primero que tenemos que hacer es registrar nuestra aplicación móvil a través de la consola de **Google Developers**. Para nuestro proyecto en particular la hemos llamado **ArmadilloAmarilloIntranet**. En el menú superior accedemos a información del proyecto dónde podemos encontrar el ID del proyecto.

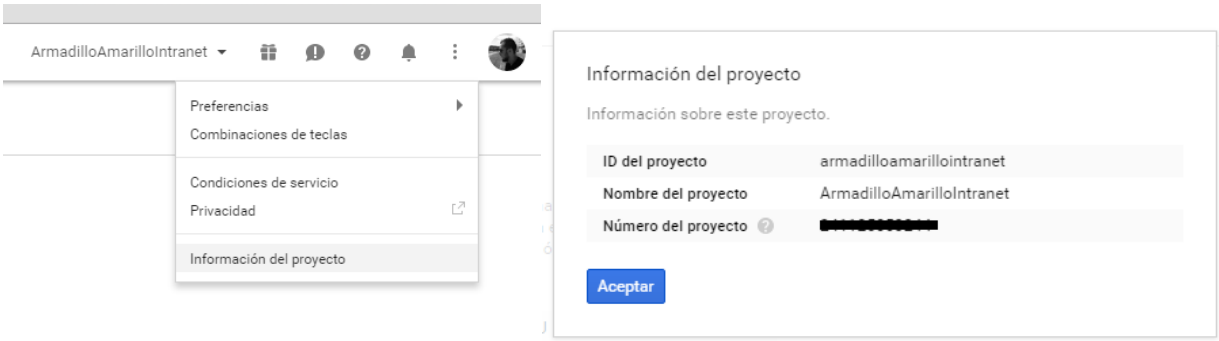

Ilustración 53 – Menú superior Android Developer Console e información del proyecto.

Existe una manera sencilla de crear un archivo de configuración para nuestro proyecto. Desde developers.google.com accedemos a la sección de Google Cloud Messaging y entramos dentro de "Get a configuration file" para llegar a la siguiente pantalla:

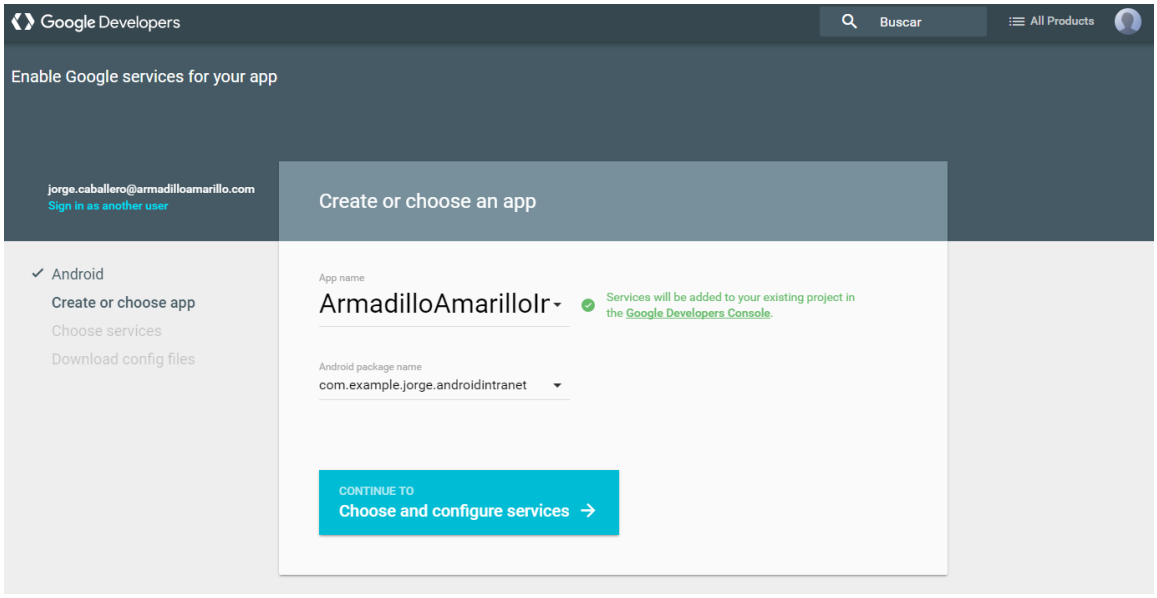

Ilustración 54 – Google Developer Console. Habilitar google services

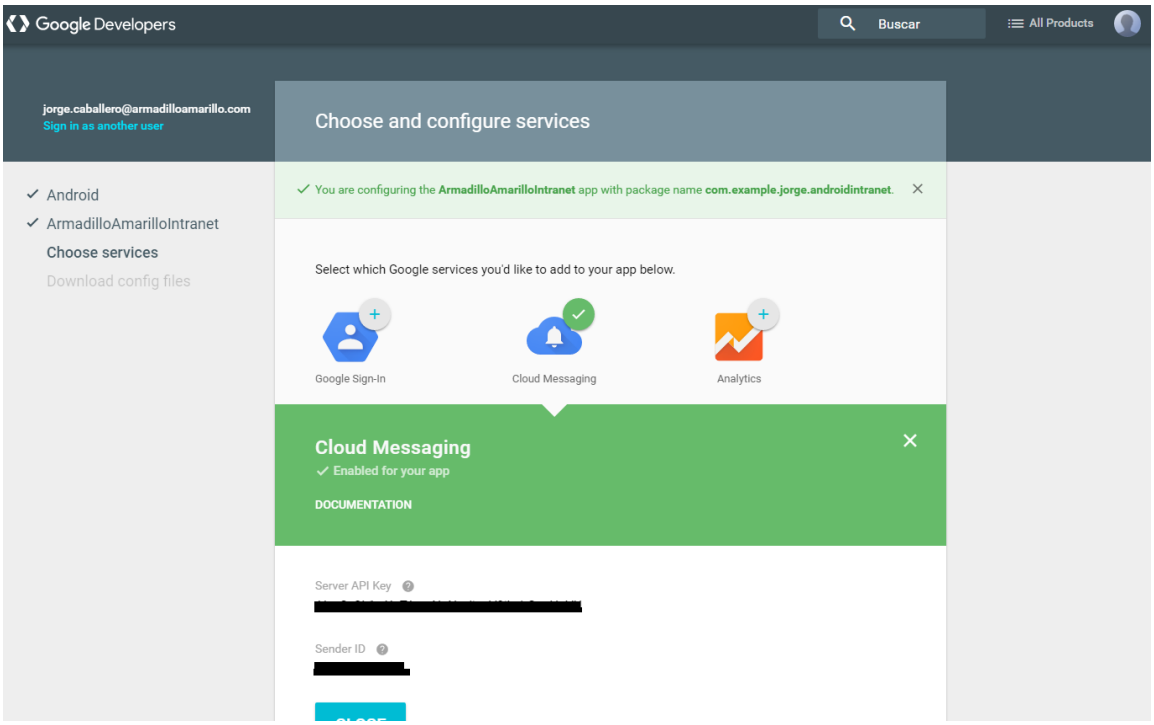

Ilustración 55 – Google Developer Console. Activar Cloud Messaging

Activamos **Cloud Messaging** y se nos genera automáticamente una clave de acceso que podemos encontrar en la consola de Developer de Google, si le damos a siguiente podemos descargar un fichero de configuración. Copiaremos google-sevices.json a nuestro proyecto en la carpeta app/ y realizamos algunas modificaciones en nuestro Gradle y en el AndroidManifest.xml.

Como curiosidad, vemos que desde este portal podemos acceder también a muchas otras funciones de Google como *son Google Analytics, Sign-in o Admob*.

Anexo VII. Diagramas relacionales de la base de datos

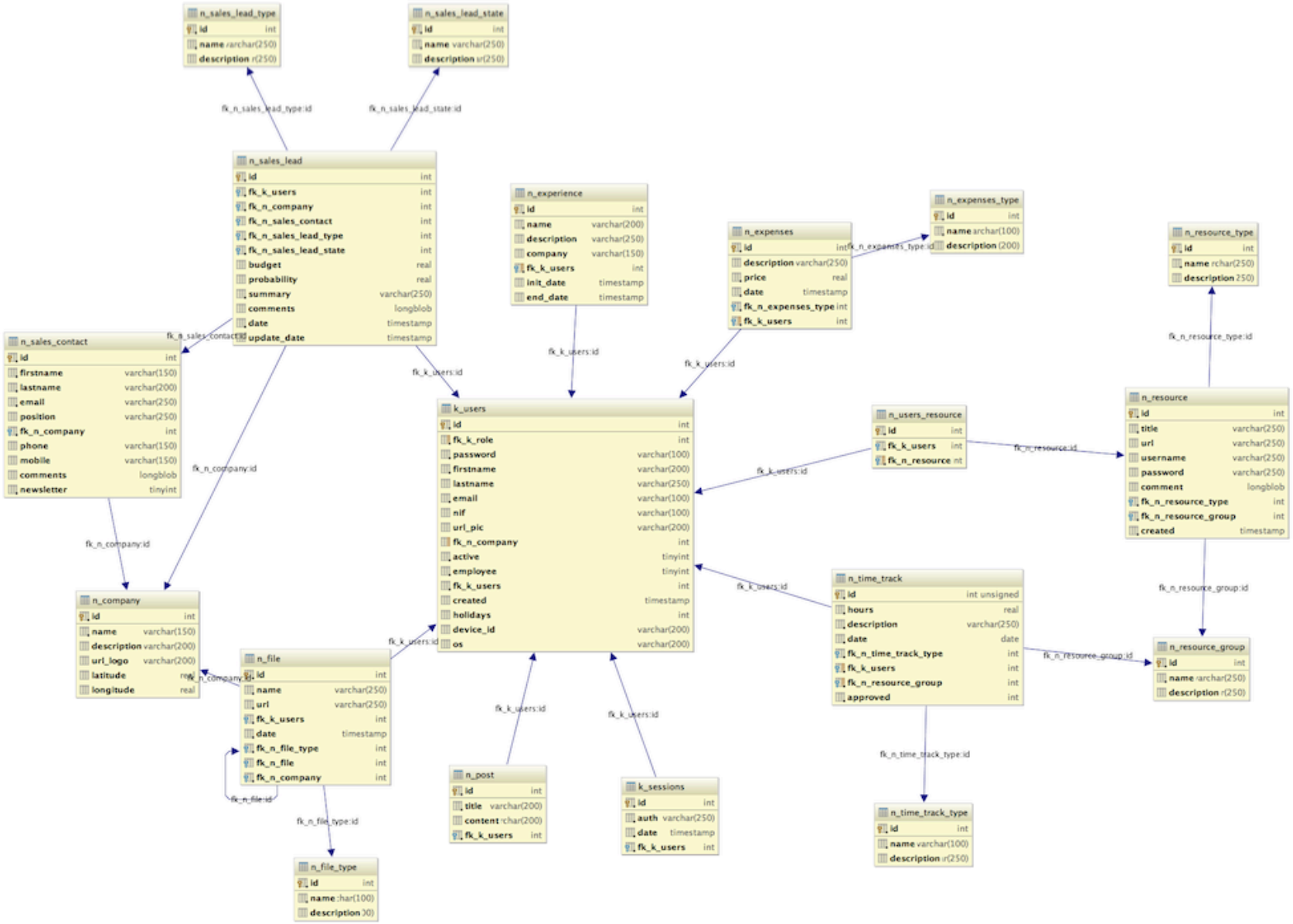

Ilustración 56 – Diagrama relacional de la base de datos (1)

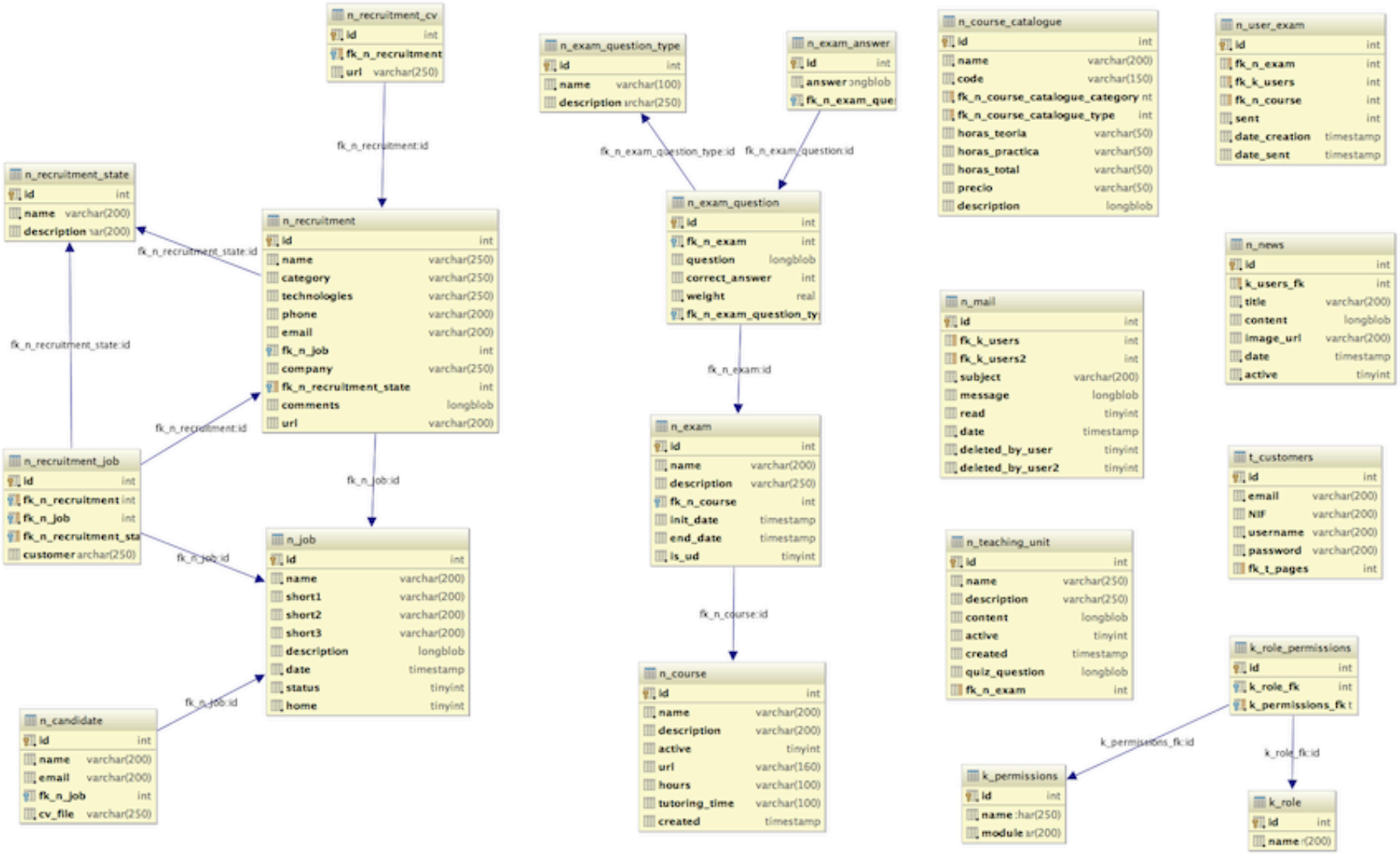

Ilustración 57 – Diagrama relacional de la base de datos (2)

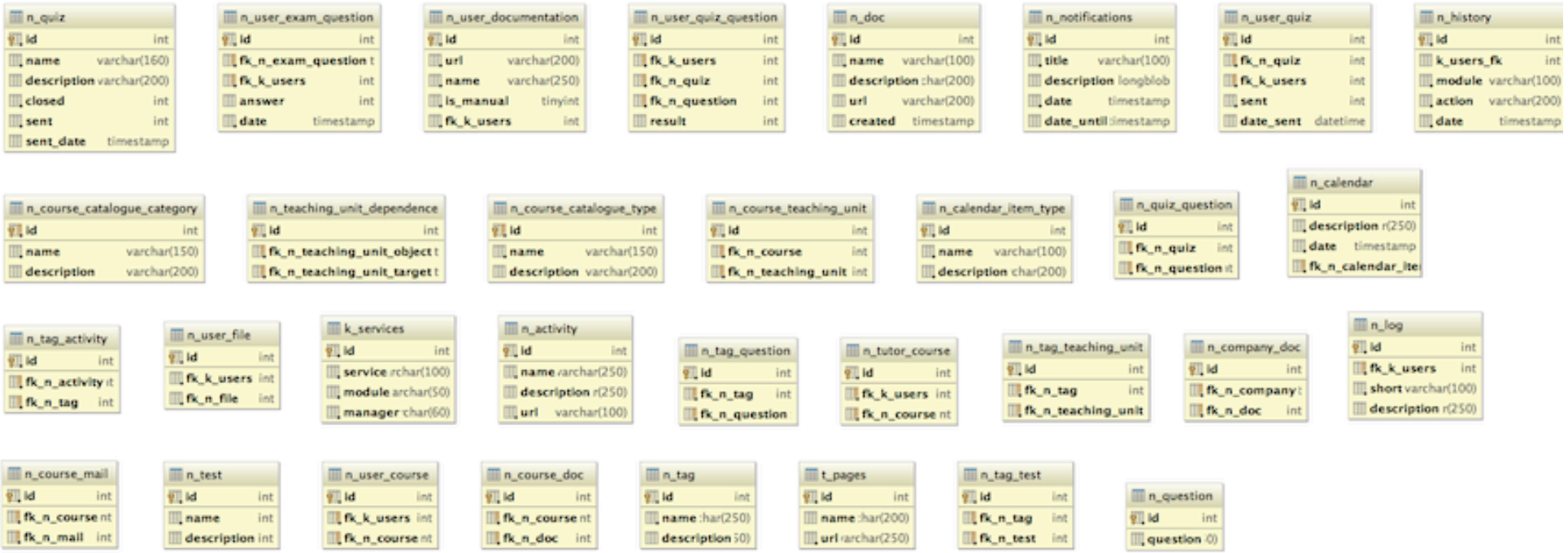

Ilustración 58 – Diagrama relacional de la base de datos (3)[19]

[19] Diagramas relacionales generados con IntelliJ 15